## Contents

1 Meteorology..............................................................................................................................1

1.1 WRF 3.8.1 Porting Guide (CentOS 7.6)......................................................................................2

1.1.1 Introduction..........................................................................................................................2

1.1.2 Environment Requirements................................................................................................2

1.1.3 Planning the Paths for Software Porting.............................................................................3

1.1.4 Configuring the Compilation Environment..........................................................................4

1.1.4.1 Setting Up the Basic Environment..................................................................................5

1.1.4.2 Installing HDF5................................................................................................................5

1.1.4.3 Installing PnetCDF...........................................................................................................6

1.1.4.4 Installing NETCDF-C......................................................................................................5

1.1.4.5 Installing NETCDF-FORTRAN.......................................................................................7

1.1.4.6 Installing OPTIMIZED-ROUTINES...............................................................................8

1.1.4.7 Installing ClusterShell....................................................................................................8

1.1.5 Obtaining Source Code........................................................................................................9

1.1.6 Compiling and Installing WRF............................................................................................9

1.1.7 Running and Verifying WRF...............................................................................................11

1.2 CAMx 6.50 Porting Guide (CentOS 7.6)..................................................................................14

1.2.1 Introduction..........................................................................................................................14

1.2.2 Environment Requirements................................................................................................14

1.2.3 Planning the Paths for Software Porting.............................................................................15

1.2.4 Configuring the Compilation Environment..........................................................................16

1.2.5 Obtaining the Source Code................................................................................................16

1.2.6 Compiling and Installing CAMx.........................................................................................17

1.2.7 Running and Verifying BWA.............................................................................................18

1.2.8 More Information................................................................................................................19

1.3 CESM 2.1.1 Porting Guide (CentOS 7.6)..................................................................................19

1.3.1 Introduction..........................................................................................................................19

1.3.2 Environment Requirements................................................................................................19

1.3.3 Planning the Paths for Software Porting.............................................................................20

1.3.4 Configuring the Compilation Environment..........................................................................21

1.3.4.1 Installing OpenBLAS......................................................................................................22

1.3.4.2 Installing HDF5................................................................................................................22

1.3.4.3 Installing PnetCDF...........................................................................................................23
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6.4.3 Installing NetCDF-Fortran</td>
<td>55</td>
</tr>
<tr>
<td>1.6.4.4 Installing ParMETIS</td>
<td>56</td>
</tr>
<tr>
<td>1.6.5 Obtaining the Source Code</td>
<td>57</td>
</tr>
<tr>
<td>1.6.6 Compiling and Installing the Software</td>
<td>57</td>
</tr>
<tr>
<td>1.6.7 Running and Verifying the Software</td>
<td>58</td>
</tr>
<tr>
<td>1.6.8 Support and Other Resources</td>
<td>60</td>
</tr>
<tr>
<td>1.7 Bifrost V1.1.1 Porting Guide (CentOS 7.6)</td>
<td>60</td>
</tr>
<tr>
<td>1.7.1 Introduction</td>
<td>60</td>
</tr>
<tr>
<td>1.7.2 Environment Requirements</td>
<td>61</td>
</tr>
<tr>
<td>1.7.3 Planning the Path for Software Porting</td>
<td>61</td>
</tr>
<tr>
<td>1.7.4 Configuring the Compilation Environment</td>
<td>62</td>
</tr>
<tr>
<td>1.7.5 Obtaining the Source Code</td>
<td>62</td>
</tr>
<tr>
<td>1.7.6 Compiling and Installing Bifrost</td>
<td>63</td>
</tr>
<tr>
<td>1.7.7 Running and Verifying Bifrost</td>
<td>64</td>
</tr>
<tr>
<td>1.8 FVCOM 4.1 Porting Guide (CentOS 7.6)</td>
<td>65</td>
</tr>
<tr>
<td>1.8.1 Introduction</td>
<td>65</td>
</tr>
<tr>
<td>1.8.2 Environment Requirements</td>
<td>66</td>
</tr>
<tr>
<td>1.8.3 Planning the Paths for Software Porting</td>
<td>67</td>
</tr>
<tr>
<td>1.8.4 Configuring the Compilation Environment</td>
<td>67</td>
</tr>
<tr>
<td>1.8.4.1 Installing HDF5</td>
<td>68</td>
</tr>
<tr>
<td>1.8.4.2 Installing NetCDF-C</td>
<td>69</td>
</tr>
<tr>
<td>1.8.4.3 Installing NetCDF-Fortran</td>
<td>69</td>
</tr>
<tr>
<td>1.8.5 Obtaining the Source Code</td>
<td>70</td>
</tr>
<tr>
<td>1.8.6 Compiling and Installing FVCOM</td>
<td>70</td>
</tr>
<tr>
<td>1.8.7 Running and Verifying FVCOM</td>
<td>73</td>
</tr>
<tr>
<td>1.8.7.1 Running and Verifying the Estuary Case</td>
<td>73</td>
</tr>
<tr>
<td>1.8.7.2 Running and Verifying the Inlet Case</td>
<td>74</td>
</tr>
<tr>
<td>1.8.8 More Information</td>
<td>76</td>
</tr>
<tr>
<td>1.9 MITGCM c67o Porting Guide (CentOS 7.6)</td>
<td>76</td>
</tr>
<tr>
<td>1.9.1 Introduction</td>
<td>76</td>
</tr>
<tr>
<td>1.9.2 Environment Requirements</td>
<td>76</td>
</tr>
<tr>
<td>1.9.3 Paths for Software Porting</td>
<td>77</td>
</tr>
<tr>
<td>1.9.4 Configuring the Compilation Environment</td>
<td>78</td>
</tr>
<tr>
<td>1.9.4.1 Installing HDF5</td>
<td>78</td>
</tr>
<tr>
<td>1.9.4.2 Installing NetCDF-C</td>
<td>79</td>
</tr>
<tr>
<td>1.9.4.3 Installing NetCDF-Fortran</td>
<td>79</td>
</tr>
<tr>
<td>1.9.5 Obtaining the Source Code</td>
<td>80</td>
</tr>
<tr>
<td>1.9.6 Compiling and Installing MITgcm</td>
<td>80</td>
</tr>
<tr>
<td>1.9.7 Running and Verifying MITgcm</td>
<td>82</td>
</tr>
<tr>
<td>1.9.8 More Information</td>
<td>83</td>
</tr>
<tr>
<td>1.10 POM POM2K Porting Guide (CentOS 7.6)</td>
<td>83</td>
</tr>
<tr>
<td>1.10.1 Introduction</td>
<td>83</td>
</tr>
</tbody>
</table>
### Contents

1.13.4 Configuring the Compilation Environment ................................................................. 111
1.13.5 Obtaining the Source Code ......................................................................................... 111
1.13.6 Compiling and Installing SPECFEM3D GLOBE .................................................. 111
1.13.7 Running and Verifying SPECFEM3D GLOBE ...................................................... 112
1.14 CMAQ 5.3.1 Porting Guide (CentOS 7.6) ................................................................. 113
1.14.1 Introduction ............................................................................................................. 113
1.14.2 Environment Requirements ..................................................................................... 113
1.14.3 Planning the Paths for Software Porting .............................................................. 115
1.14.4 Configuring the Compilation Environment .......................................................... 115
1.14.4.1 Installing HDF5 ............................................................................................... 116
1.14.4.2 Installing NetCDF-C ....................................................................................... 116
1.14.4.3 Installing NetCDF-Fortran ............................................................................ 117
1.14.4.4 Installing PnetCDF ....................................................................................... 117
1.14.5 Obtaining the Source Code .................................................................................... 118
1.14.6 Compiling and Installing CMAQ .......................................................................... 118
1.14.7 Running and Verifying CMAQ .............................................................................. 121
1.14.8 Troubleshooting .................................................................................................... 122
1.15 SWAN 41.31 Porting Guide (CentOS 7.6) ............................................................... 122
1.15.1 Introduction ............................................................................................................ 123
1.15.2 Environment Requirements ................................................................................... 123
1.15.3 Planning the Paths for Software Porting .............................................................. 124
1.15.4 Configuring the Compilation Environment .......................................................... 125
1.15.4.1 Installing HDF5 ............................................................................................... 126
1.15.4.2 Installing PnetCDF ....................................................................................... 126
1.15.4.3 Installing NetCDF-C ....................................................................................... 127
1.15.4.4 Installing NetCDF-Fortran ............................................................................ 128
1.15.5 Obtaining the Source Code .................................................................................... 129
1.15.6 Compiling and Installing SWAN ......................................................................... 129
1.15.7 Running and Verifying SWAN ............................................................................. 130
1.15.8 More Information .................................................................................................. 131
1.16 CDO 1.9.8 Porting Guide (CentOS 7.6) ................................................................. 131
1.16.1 Introduction ........................................................................................................... 131
1.16.2 Environment Requirements ................................................................................... 132
1.16.3 Paths for Software Porting .................................................................................... 133
1.16.4 Configuring the Compilation Environment .......................................................... 133
1.16.4.1 Setting Up the Basic Environment ................................................................ 134
1.16.4.2 Configuring the Huawei Yum Source ............................................................ 134
1.16.4.2.1 Procedure .................................................................................................. 134
1.16.4.3 Installing the Dependency Package ............................................................. 134
1.16.4.3.1 Procedure .................................................................................................. 134
1.16.5 Obtaining Source Code ....................................................................................... 135
1.16.6 Compilation and Installation ............................................................................... 135
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.16.7</td>
<td>Running and Verifying CDO</td>
<td>135</td>
</tr>
<tr>
<td>1.16.8</td>
<td>More Resources</td>
<td>136</td>
</tr>
<tr>
<td>1.17</td>
<td>ecCodes 2.18.0 Porting Guide (CentOS 7.6)</td>
<td>136</td>
</tr>
<tr>
<td>1.17.1</td>
<td>Introduction</td>
<td>136</td>
</tr>
<tr>
<td>1.17.2</td>
<td>Environment Requirements</td>
<td>137</td>
</tr>
<tr>
<td>1.17.3</td>
<td>Paths for Software Porting</td>
<td>138</td>
</tr>
<tr>
<td>1.17.4</td>
<td>Configuring the Compilation Environment</td>
<td>138</td>
</tr>
<tr>
<td>1.17.4.1</td>
<td>Installing CMake</td>
<td>139</td>
</tr>
<tr>
<td>1.17.4.1.1</td>
<td>Procedure</td>
<td>139</td>
</tr>
<tr>
<td>1.17.5</td>
<td>Obtaining Source Code</td>
<td>139</td>
</tr>
<tr>
<td>1.17.6</td>
<td>Compilation and Installation</td>
<td>139</td>
</tr>
<tr>
<td>1.17.7</td>
<td>Running and Verifying ecCodes</td>
<td>140</td>
</tr>
<tr>
<td>1.17.8</td>
<td>More Resources</td>
<td>140</td>
</tr>
<tr>
<td>1.18</td>
<td>ecFlow 5.5.2 Porting Guide (CentOS 7.6)</td>
<td>141</td>
</tr>
<tr>
<td>1.18.1</td>
<td>Introduction</td>
<td>141</td>
</tr>
<tr>
<td>1.18.2</td>
<td>Environment Requirements</td>
<td>141</td>
</tr>
<tr>
<td>1.18.3</td>
<td>Paths for Software Porting</td>
<td>142</td>
</tr>
<tr>
<td>1.18.4</td>
<td>Configuring the Compilation Environment</td>
<td>143</td>
</tr>
<tr>
<td>1.18.4.1</td>
<td>Installing CMake</td>
<td>143</td>
</tr>
<tr>
<td>1.18.5</td>
<td>Obtaining Source Code</td>
<td>144</td>
</tr>
<tr>
<td>1.18.6</td>
<td>Compilation and Installation</td>
<td>144</td>
</tr>
<tr>
<td>1.18.7</td>
<td>Running and Verifying ecFlow</td>
<td>145</td>
</tr>
<tr>
<td>1.18.8</td>
<td>More Resources</td>
<td>146</td>
</tr>
<tr>
<td>1.19</td>
<td>NCEPLIBS 1.2.0 Porting Guide (CentOS 7.6)</td>
<td>146</td>
</tr>
<tr>
<td>1.19.1</td>
<td>Introduction</td>
<td>146</td>
</tr>
<tr>
<td>1.19.2</td>
<td>Environment Requirements</td>
<td>146</td>
</tr>
<tr>
<td>1.19.3</td>
<td>Paths for Software Porting</td>
<td>147</td>
</tr>
<tr>
<td>1.19.4</td>
<td>Configuring the Compilation Environment</td>
<td>148</td>
</tr>
<tr>
<td>1.19.4.1</td>
<td>Installing HDF5</td>
<td>149</td>
</tr>
<tr>
<td>1.19.4.2</td>
<td>Installing PNETCDF</td>
<td>149</td>
</tr>
<tr>
<td>1.19.4.3</td>
<td>Installing NETCDF</td>
<td>150</td>
</tr>
<tr>
<td>1.19.4.4</td>
<td>Installing NETCDF-FORTRAN</td>
<td>150</td>
</tr>
<tr>
<td>1.19.4.5</td>
<td>Installing CMake</td>
<td>151</td>
</tr>
<tr>
<td>1.19.5</td>
<td>Obtaining Source Code</td>
<td>151</td>
</tr>
<tr>
<td>1.19.6</td>
<td>Compilation and Installation</td>
<td>151</td>
</tr>
<tr>
<td>1.19.7</td>
<td>Running and Verifying NCEPLIBS</td>
<td>152</td>
</tr>
<tr>
<td>1.19.8</td>
<td>Troubleshooting</td>
<td>153</td>
</tr>
<tr>
<td>1.19.9</td>
<td>More Resources</td>
<td>153</td>
</tr>
<tr>
<td>1.20</td>
<td>Pyferret 7.6.0 Porting Guide (CentOS 7.6)</td>
<td>153</td>
</tr>
<tr>
<td>1.20.1</td>
<td>Introduction</td>
<td>154</td>
</tr>
<tr>
<td>1.20.2</td>
<td>Environment Requirements</td>
<td>154</td>
</tr>
<tr>
<td>1.20.3</td>
<td>Paths for Software Porting</td>
<td>155</td>
</tr>
</tbody>
</table>
2 Manufacturing ........................................................................................................................................... 168

1.20.4 Configuring the Compilation Environment ................................................................................................. 155
1.20.4.1 Setting Up the Basic Environment .................................................................................................................. 156
1.20.4.2 Configuring the Yum Source and Installing the Dependencies ................................................................. 156
1.20.4.3 Installing HDF5 ............................................................................................................................................... 157
1.20.4.4 Installing NETCDF ........................................................................................................................................... 157
1.20.4.5 Installing NETCDF-FORTRAN ....................................................................................................................... 158
1.20.5 Obtaining Source Code ...................................................................................................................................... 158
1.20.6 Compilation and Installation .............................................................................................................................. 159
1.20.7 Running and Verifying Pyferret ....................................................................................................................... 160
1.20.8 More Resources ................................................................................................................................................ 162
1.21 ncl 6.3.0 Porting Guide (CentOS 7.6) ................................................................................................................... 162
1.21.1 Introduction ........................................................................................................................................................ 162
1.21.2 Environment Requirements ................................................................................................................................. 163
1.21.3 Paths for Software Porting .................................................................................................................................. 163
1.21.4 Configuring the Compilation Environment ....................................................................................................... 164
1.21.5 Compilation and Installation ............................................................................................................................... 164
1.21.6 Running and Verifying ncl .................................................................................................................................. 167

2.1 OpenFOAM 1906 Porting Guide (CentOS 7.6) ........................................................................................................ 168
2.1.1 Introduction .......................................................................................................................................................... 168
2.1.2 Environment Requirements ................................................................................................................................. 169
2.1.3 Planning the Paths for Software Porting ............................................................................................................... 169
2.1.4 Configuring the Compilation Environment ....................................................................................................... 170
2.1.4.1 Installing ClusterShell .......................................................................................................................................... 170
2.1.5 Obtaining the Source Code .................................................................................................................................. 171
2.1.6 Compiling and Installing OpenFOAM .................................................................................................................... 171
2.1.7 Running and Verifying OpenFOAM ....................................................................................................................... 172
2.1.8 Troubleshooting .................................................................................................................................................... 175
2.2 SU2 Porting Guide (CentOS 7.6) .............................................................................................................................. 177
2.2.1 Introduction .......................................................................................................................................................... 177
2.2.2 Environment Requirements .................................................................................................................................. 177
2.2.3 Planning the Paths for Software Porting ............................................................................................................... 178
2.2.4 Configuring the Compilation Environment ....................................................................................................... 179
2.2.4.1 Installing HDF5 .................................................................................................................................................... 179
2.2.5 SU2 6.0.1 Porting Guide (CentOS 7.6) ..................................................................................................................... 180
2.2.5.1 Obtaining the Source Code .................................................................................................................................. 180
2.2.5.2 Compiling and Installing SU2 ............................................................................................................................. 180
2.2.6 SU2 7.0.4 Porting Guide (CentOS 7.6) ..................................................................................................................... 181
2.2.6.1 Obtaining the Source Code .................................................................................................................................. 181
2.2.6.2 Compiling and Installing SU2 ............................................................................................................................. 181
2.2.7 Running and Verifying SU2 .................................................................................................................................. 182
2.2.8 Troubleshooting .................................................................................................................................................... 184
2.9.4.1 Running and Verifying Code_Saturne on a Single Node.......................................................................................................................... 206
2.9.4.2 Running and Verifying Code_Saturne on Multiple Nodes................................................................................................................ 207
2.9.5 More Information.................................................................................................................................................................................. 209

2.8.1 Introduction................................................................................................................................................................................. 214
2.8.2 Environment Requirements......................................................................................................................................................... 215
2.8.3 Planning the Paths for Software Porting........................................................................................................................................ 216
2.8.4 Configuring the Compilation Environment........................................................................................................................................ 216
2.8.4.1 Installing CMake.................................................................................................................................................................. 217
2.8.4.2 Installing OpenBLAS.......................................................................................................................................................... 217
2.8.5 Obtaining the Source Code......................................................................................................................................................... 218
2.8.6 Compiling and Installing Elmer................................................................................................................................................ 218
2.8.7 Running and Verifying Elmer..................................................................................................................................................... 219
2.8.8 Troubleshooting....................................................................................................................................................................... 220
2.8.9 More Information....................................................................................................................................................................... 220

2.7.2 Environment Requirements......................................................................................................................................................... 210
2.7.3 Planning the Paths for Software Porting........................................................................................................................................ 210
2.7.4 Configuring the Compilation Environment........................................................................................................................................ 211
2.7.5 Obtaining the Source Code......................................................................................................................................................... 211
2.7.6 Compiling and Installing FDS.................................................................................................................................................... 212
2.7.7 Running and Verifying FDS..................................................................................................................................................... 212
2.7.8 Troubleshooting....................................................................................................................................................................... 214
2.7.9 More Information....................................................................................................................................................................... 214

2.6.2.2 Installing OpenBLAS.......................................................................................................................................................... 205
2.6.3 More Information........................................................................................................................................................................ 209

2.5.7.1 Running and Verifying Code_Saturne on a Single Node.......................................................................................................................... 206
2.5.7.2 Running and Verifying Code_Saturne on Multiple Nodes................................................................................................................ 207
2.5.8 More Information........................................................................................................................................................................ 209

2.4.1.1 Installing CMake.................................................................................................................................................................. 205
2.4.2.1 Installing OpenBLAS.......................................................................................................................................................... 205
2.4.3 More Information........................................................................................................................................................................ 209

2.3.2 Environment Requirements......................................................................................................................................................... 204
2.3.3 Planning the Paths for Software Porting........................................................................................................................................ 204
2.3.4 Configuring the Compilation Environment........................................................................................................................................ 204
2.3.5 Obtaining the Source Code......................................................................................................................................................... 204
2.3.6 Compiling and Installing Code_Saturne........................................................................................................................................ 205
2.3.7 Running and Verifying Code_Saturne on a Single Node.......................................................................................................................... 205
2.3.8 Running and Verifying Code_Saturne on Multiple Nodes................................................................................................................ 206
2.3.9 More Information........................................................................................................................................................................ 209

2.2.6.1 Installing OpenBLAS.......................................................................................................................................................... 203
2.2.7 More Information........................................................................................................................................................................ 207

2.1.2 Environment Requirements......................................................................................................................................................... 201
2.1.3 Planning the Paths for Software Porting........................................................................................................................................ 202
2.1.4 Configuring the Compilation Environment........................................................................................................................................ 202
2.1.5 Obtaining the Source Code......................................................................................................................................................... 202
2.1.6 Compiling and Installing Code_Aster........................................................................................................................................ 203
2.1.7 Running and Verifying Code_Aster........................................................................................................................................... 203
2.1.8 Troubleshooting....................................................................................................................................................................... 204
2.1.9 More Information....................................................................................................................................................................... 204

1.1.2 Environment Requirements......................................................................................................................................................... 195
1.1.3 Planning the Paths for Software Porting........................................................................................................................................ 195
1.1.4 Configuring the Compilation Environment........................................................................................................................................ 195
1.1.5 Obtaining the Source Code......................................................................................................................................................... 195
1.1.6 Compiling and Installing Code_Aster........................................................................................................................................ 196
1.1.7 Running and Verifying Code_Aster........................................................................................................................................... 196
1.1.8 Troubleshooting....................................................................................................................................................................... 197
1.1.9 More Information....................................................................................................................................................................... 197

3 Government HPC........................................................................................................................................................................... 230
3.1 udunits 2.2.28 Porting Guide (CentOS 7.6)........................................................................................................................................ 231
3.1.1 Introduction............................................................................................................................................................................. 231
3.1.2 Environment Requirements.................................................................................................................................................... 231
3.3.8 More Information............................................................................................................................................................245
3.3.7 Running and Verifying GROMACS............................................................................................................................ 244
3.3.6 Compiling and Installing GROMACS........................................................................................................................ 243
3.3.5 Obtaining the Source Code......................................................................................................................................... 243
3.3.4 Planning the Paths for Software Porting................................................................................................................ 240
3.3.3 Planning the Paths for Software Porting................................................................................................................ 239
3.3.2 Environment Requirements......................................................................................................................................... 239
3.3.1 Introduction.......................................................................................................................................................................238
3.3 GROMACS 2019.3 Porting Guide (CentOS 7.6)....................................................................................................... 238
3.2.7 Running and Verifying CANU..................................................................................................................................... 237
3.2.5 Obtaining the Source Code......................................................................................................................................... 236
3.2.4 Compiling and Installing CANU.............................................................................................................................. 236
3.2.3 Planning the Paths for Software Porting................................................................................................................ 235
3.2.2 Environment Requirements......................................................................................................................................... 234
3.2.1 Introduction.......................................................................................................................................................................234
3.2 CANU 1.8 Porting Guide (CentOS 7.6)................................................................................................................................ 234
3.1.7 Running and Verifying UDUNITS...............................................................................................................................234
3.1.6 Compiling and Installing UDUNITS.......................................................................................................................... 233
3.1.5 Obtaining the Source Code......................................................................................................................................... 233
3.1.4 Planning the Paths for Software Porting................................................................................................................ 232
3.1.3 Planning Paths for Software Porting........................................................................................................................ 232
3.1.2 Configuring the Compilation Environment............................................................................................................ 233
3.1.1 Introduction.......................................................................................................................................................................232
3.1 UDUNIT 1.2 Porting Guide (CentOS 7.6).................................................................................................................... 232
3.0 Porting Guide
Kunpeng BoostKit for HPC
Issue 14 (2021-07-23)
Copyright © Huawei Technologies Co., Ltd.
3.8.8 Troubleshooting ............................................................................................................................................................... 282
3.8.7 Running and Verifying Ceph ....................................................................................................................................... 280
3.8.6 Compiling and Installing ABINIT ............................................................................................................................... 279
3.8.5 Obtaining the Source Code ......................................................................................................................................... 279
3.8.4.4 Installing NetCDF-C .................................................................................................................................................... 277
3.8.4.3 Installing HDF5 .............................................................................................................................................................277
3.8.4.1 Installing OpenBLAS ................................................................................................................................................... 276
3.8.3 Planning the Paths for Software Porting ................................................................................................................ 274
3.8.2 Environment Requirements ......................................................................................................................................... 273
3.8.1 Introduction .......................................................................................................................................................................273
3.7.8 Support and Other Resources.....................................................................................................................................272
3.7.7 Running and Verifying the Software........................................................................................................................ 271
3.7.6.2.2 Installing the Wannier90 Interfaces...................................................................................................................270
3.7.6.2.1 Installing the VTST Transition State Tools ....................................................................................................... 269
3.7.6.2 (Optional) Extended Installation ........................................................................................................................... 269
3.7.6.1 Compiling and Installing Quantum ESPRESSO .....................................................................................................254
3.7.5 Obtaining the Source Code ......................................................................................................................................... 267
3.7.4.3 Installing FFTW .............................................................................................................................................................267
3.7.4.2 Installing ScaLapack ................................................................................................................................................... 266
3.7.4 Installing FFTW .................................................................................................................................................................266
3.7.3 Planning the Paths for Software Porting ................................................................................................................ 264
3.7.2 Environment Requirements ......................................................................................................................................... 263
3.7.1 Introduction .......................................................................................................................................................................263
3.7 VASP 5.4.4 Porting Guide (CentOS 7.6) ......................................................................................................................263
3.6.8 More Information ............................................................................................................................................................263
3.6.7 Running and Verifying NAMD.................................................................................................................................... 262
3.6.6 Compiling and Installing NAMD................................................................................................................................260
3.6.5 Obtaining the Source Code ......................................................................................................................................... 260
3.6.4.1 Installing OpenBLAS ................................................................................................................................................... 259
3.6.4.2 Installing Pnetcdf ...........................................................................................................................................................259
3.6.4.3 Installing HDF5.............................................................................................................................................................259
3.6.4 Installing NetCDF-C ..........................................................................................................................................................259
3.6.3 Planning the Paths for Software Porting ................................................................................................................ 258
3.6.2 Environment Requirements ......................................................................................................................................... 257
3.6.1 Introduction .......................................................................................................................................................................257
3.6 NAMD 2.13 Porting Guide (CentOS 7.6) ........................................................................................................................ 255
3.6.6 Compiling and Installing ABINIT ................................................................................................................................254
3.6.5 Obtaining the Source Code ......................................................................................................................................... 254
3.6.4 Installing NetCDF-Fortran ........................................................................................................................................... 254
3.6.3 Planning the Paths for Software Porting ................................................................................................................ 254
3.6.2 Environment Requirements ......................................................................................................................................... 254
3.6.1 Introduction .......................................................................................................................................................................254
3.5.6 Compiling and Installing Quantum ESPRESSO .....................................................................................................254
3.11.4.11 Installing GSL
3.11.4.9 Installing Spglib
3.11.4.8 Installing ELPA
3.11.4.7 Installing ScaLAPACK
3.11.4.5 Installing FFTW
3.11.4.4 Installing Libint
3.11.4.3 Installing Boost
3.11.4.2 Installing GMP
3.11.4.1 Installing CMake
### Contents

- 3.12 NWChem 6.8.1 Porting Guide (CentOS 7.6) ................................................................. 310
- 3.11.8 More Information .................................................................................................. 310
- 3.15.1 Introduction ........................................................................................................ 328
- 3.15 Ont-Tombo 1.5.1 Porting Guide (CentOS 7.6) ........................................................... 328
- 3.14.9 More Information .................................................................................................. 328
- 3.14.7 Running and Verifying Geant4 ............................................................................. 327
- 3.14.5 Obtaining the Source Code .................................................................................. 325
- 3.14.4 Environment Requirements ................................................................................ 322
- 3.14.2 Environment Requirements ................................................................................ 322
- 3.14.1 Introduction ........................................................................................................ 322
- 3.14 Geant4 10.6 Porting Guide (CentOS 7.6) ................................................................. 322
- 3.13.8 Troubleshooting .................................................................................................. 321
- 3.13.7 Running and Verifying the ROOT ....................................................................... 319
- 3.13.6 Compiling and Installing NEMO ........................................................................ 319
- 3.13.5 Obtaining the Source Code .................................................................................. 318
- 3.13.4 Planning the Paths for Software Porting ............................................................... 316
- 3.13.3 Planning the Paths for Software Porting ............................................................... 316
- 3.13.2 Environment Requirements ................................................................................ 315
- 3.13.1 Introduction ........................................................................................................ 315
- 3.13 ROOT 6.20 Porting Guide (CentOS 7.6) .................................................................. 315
- 3.12.9 More Information .................................................................................................. 315
- 3.12.8 Troubleshooting .................................................................................................. 315
- 3.12.7 Running and Verifying BWA ............................................................................... 313
- 3.12.6 Compiling and Installing NWChem ................................................................. 312
- 3.12.5 Obtaining the Source Code .................................................................................. 312
- 3.12.4 Planning the Paths for Software Porting ............................................................... 311
- 3.12.3 Planning the Paths for Software Porting ............................................................... 311
- 3.12.2 Environment Requirements ................................................................................ 310
- 3.12.1 Introduction ........................................................................................................ 310
- 3.12 NWChem 6.8.1 Porting Guide (CentOS 7.6) ................................................................. 310
- 3.11.7 Running and Verifying CP2K ............................................................................. 309
- 3.11.6 Compiling and Installing CP2K ......................................................................... 308
- 3.11.5 Obtaining the Source Code .................................................................................. 308
- 3.11.4.12 Installing PLUMED ..................................................................................... 307
3.16.7.1 Running and Verifying DL_POLY on a Single Node.......................................................................................337
3.16.5 Obtaining the Source Code....................................................................................................................................... 336
3.16.4 Compiling and Installing DL_POLY.......................................................................................................................... 336
3.16.3 Planning the Paths for Software Porting.................................................................................................................. 336
3.16.2 Environment Requirements........................................................................................................................................ 334
3.16.1 Introduction.................................................................................................................................................................... 334
3.15.6 Compiling and Installing Ont-Tombo.................................................................................................................... 334
3.15.5 Obtaining the Source Code........................................................................................................................................ 333
3.15.4.5 Installing HDF5.......................................................................................................................................................... 333
3.15.4.4 Installing OpenBLAS.................................................................................................................................................332
3.15.4.3 Installing PIP............................................................................................................................................................... 332
3.15.4.2 Installing Python........................................................................................................................................................331
3.15.4.1 Installing OpenBLAS.................................................................................................................................................331
3.15.4 Installing HDF5............................................................................................................................................................ 331
3.15.3 Planning the Paths for Software Porting.................................................................................................................. 330
3.15.2 Environment Requirements.......................................................................................................................................330
3.15.1 Introduction.................................................................................................................................................................... 330
3.14.8 More Resources............................................................................................................................................................. 329
3.14.7 Running and Verifying GAMESS.............................................................................................................................. 329
3.14.6 Obtaining the Source Code........................................................................................................................................ 328
3.14.5 Configuring the Compilation Environment................................................................................................................. 328
3.14.4 Compiling and Installing GAMESS......................................................................................................................... 328
3.14.3 Planning the Paths for Software Porting.................................................................................................................. 326
3.14.2 Environment Requirements.......................................................................................................................................326
3.14.1 Introduction.................................................................................................................................................................... 326
3.13.8 More Resources............................................................................................................................................................. 325
3.13.7 Running and Verifying CPMD................................................................................................................................... 325
3.13.6 Obtaining the Source Code........................................................................................................................................ 324
3.13.5 Configuring the Compilation Environment................................................................................................................. 324
3.13.4 Compiling and Installing CPMD................................................................................................................................. 324
3.13.3 Planning the Paths for Software Porting.................................................................................................................. 322
3.13.2 Environment Requirements.......................................................................................................................................322
3.13.1 Introduction.................................................................................................................................................................... 322
3.12.8 More Resources............................................................................................................................................................. 321
3.12.7 Running and Verifying SASSB................................................................................................................................... 321
3.12.6 Obtaining the Source Code........................................................................................................................................ 320
3.12.5 Configuring the Compilation Environment................................................................................................................. 320
3.12.4 Compiling and Installing SASSB................................................................................................................................. 320
3.12.3 Planning the Paths for Software Porting.................................................................................................................. 318
3.12.2 Environment Requirements.......................................................................................................................................318
3.12.1 Introduction.................................................................................................................................................................... 318
3.11.8 More Resources............................................................................................................................................................. 317
3.11.7 Running and Verifying GROMACS............................................................................................................................. 317
3.11.6 Obtaining the Source Code........................................................................................................................................ 316
3.11.5 Configuring the Compilation Environment................................................................................................................. 316
3.11.4 Compiling and Installing GROMACS........................................................................................................................ 316
3.11.3 Planning the Paths for Software Porting.................................................................................................................. 314
3.11.2 Environment Requirements.......................................................................................................................................314
3.11.1 Introduction.................................................................................................................................................................... 314
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.22.2 Environment Requirements</td>
<td>374</td>
</tr>
<tr>
<td>3.22.3 Paths for Software Porting</td>
<td>375</td>
</tr>
<tr>
<td>3.22.4 Configuring the Compilation Environment</td>
<td>376</td>
</tr>
<tr>
<td>3.22.4.1 Installing Dependencies</td>
<td>376</td>
</tr>
<tr>
<td>3.22.4.2 Installing FFTW</td>
<td>376</td>
</tr>
<tr>
<td>3.22.4.3 Installing GSL</td>
<td>377</td>
</tr>
<tr>
<td>3.22.4.4 Installing Libxc</td>
<td>377</td>
</tr>
<tr>
<td>3.22.5 Obtaining the Source Code</td>
<td>378</td>
</tr>
<tr>
<td>3.22.6 Compilation and Installation</td>
<td>378</td>
</tr>
<tr>
<td>3.22.7 Running and Verification</td>
<td>379</td>
</tr>
<tr>
<td>3.22.8 More Resources</td>
<td>380</td>
</tr>
<tr>
<td>3.23 Psi4 1.3.2 Porting Guide (CentOS 7.6)</td>
<td>380</td>
</tr>
<tr>
<td>3.23.1 Introduction</td>
<td>380</td>
</tr>
<tr>
<td>3.23.2 Environment Requirements</td>
<td>381</td>
</tr>
<tr>
<td>3.23.3 Paths for Software Porting</td>
<td>382</td>
</tr>
<tr>
<td>3.23.4 Configuring the Compilation Environment</td>
<td>382</td>
</tr>
<tr>
<td>3.23.4.1 Installing System Dependencies</td>
<td>383</td>
</tr>
<tr>
<td>3.23.4.2 Installing CMake</td>
<td>383</td>
</tr>
<tr>
<td>3.23.4.3 Installing OpenBLAS</td>
<td>384</td>
</tr>
<tr>
<td>3.23.4.4 Installing gau2grid</td>
<td>385</td>
</tr>
<tr>
<td>3.23.4.5 Installing Libint</td>
<td>385</td>
</tr>
<tr>
<td>3.23.4.6 Installing Libxc</td>
<td>386</td>
</tr>
<tr>
<td>3.23.4.7 Installing pybind11</td>
<td>386</td>
</tr>
<tr>
<td>3.23.5 Obtaining the Source Code</td>
<td>387</td>
</tr>
<tr>
<td>3.23.6 Compilation and Installation</td>
<td>387</td>
</tr>
<tr>
<td>3.23.7 Running and Verification</td>
<td>388</td>
</tr>
<tr>
<td>3.23.8 More Resources</td>
<td>388</td>
</tr>
<tr>
<td>3.24 SIESTA 4.0.2 Porting Guide (CentOS 7.6)</td>
<td>388</td>
</tr>
<tr>
<td>3.24.1 Introduction</td>
<td>389</td>
</tr>
<tr>
<td>3.24.2 Environment Requirements</td>
<td>389</td>
</tr>
<tr>
<td>3.24.3 Paths for Software Porting</td>
<td>390</td>
</tr>
<tr>
<td>3.24.4 Configuring the Compilation Environment</td>
<td>390</td>
</tr>
<tr>
<td>3.24.4.1 Installing Dependencies</td>
<td>391</td>
</tr>
<tr>
<td>3.24.4.2 Installing ScaLAPACK</td>
<td>391</td>
</tr>
<tr>
<td>3.24.5 Obtaining the Source Code</td>
<td>392</td>
</tr>
<tr>
<td>3.24.6 Compilation and Installation</td>
<td>392</td>
</tr>
<tr>
<td>3.24.7 Running and Verification</td>
<td>393</td>
</tr>
<tr>
<td>3.24.8 Troubleshooting</td>
<td>393</td>
</tr>
<tr>
<td>3.24.9 More Resources</td>
<td>394</td>
</tr>
<tr>
<td>3.25 Wannier90 3.1.0 Porting Guide (CentOS 7.6)</td>
<td>394</td>
</tr>
<tr>
<td>3.25.1 Introduction</td>
<td>394</td>
</tr>
<tr>
<td>3.25.2 Environment Requirements</td>
<td>394</td>
</tr>
</tbody>
</table>
3.25.3 Paths for Software Porting
3.25.4 Configuring the Compilation Environment
3.25.4.1 Installing the Dependency Libraries and Tools Using Yum
3.25.5 Obtaining the Source Code
3.25.6 Compilation and Installation
3.25.7 Running and Verification
3.25.8 More Resources

3.26 WanT 2.6.1 Porting Guide (CentOS 7.6)
3.26.1 Introduction
3.26.2 Environment Requirements
3.26.3 Paths for Software Porting
3.26.4 Configuring the Compilation Environment
3.26.4.1 Installing QE
3.26.5 Obtaining the Source Code
3.26.6 Compilation and Installation
3.26.7 Running and Verification
3.26.8 More Resources

4 Others
4.1 Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6)
4.1.1 Introduction
4.1.2 Environment Requirements
4.1.3 Planning the Paths for Software Porting
4.1.4 Configuring the Compilation Environment
4.1.4.1 Installing the Java Environment
4.1.5 Obtaining the Source Code
4.1.6 Compiling and Installing Hadoop and Spark
4.1.6.1 Configuring Hadoop
4.1.6.2 Configuring Spark
4.1.7 Running and Verifying Hadoop and Spark
4.1.8 Troubleshooting
4.1.9 More Information
4.2 SGE 8.1.9 Porting Guide (CentOS 7.6)
4.2.1 Introduction
4.2.2 Environment Requirements
4.2.3 Planning the Paths for Software Porting
4.2.4 Configuring the Compilation Environment
4.2.5 Obtaining the Source Code
4.2.6 Compiling and Installing SGE
4.2.6.1 Compiling SGE
4.2.6.2 Installing SGE on the Master Host
4.2.6.3 Installing SGE on the Execution Host
4.2.7 Running and Verifying SGE
4.22.7 Running and Verifying Velvet...................................................................................................................................559
4.22.5 Obtaining the Source Code....................................................................................................................................... 558
4.22.4 Environment Requirements.......................................................................................................................................557
4.22.2 Environment Requirements.......................................................................................................................................557
4.22.1 Introduction.................................................................................................................................................................... 556
4.22 Velvet 1.2.10 Porting Guide (CentOS 7.6)............................................................................................................... 556

4.21.7 Testing and
4.21.6 Compiling and Installing Software......................................................................................................................... 555
4.21.4
4.21.3 Planning the Paths for Software Porting..............................................................................................................553
4.21.2 Environment Requirements.......................................................................................................................................552
4.21.1 Introduction.................................................................................................................................................................... 552
4.21 pblat 2.1 Porting Guide (CentOS 7.6).......................................................................................................................547
4.20.9 More Resources............................................................................................................................................................. 552
4.20.8 Troubleshooting............................................................................................................................................................ 552
4.20.7 Running and Verifying pblat.....................................................................................................................................551
4.20.6 Compiling and Installing pblat.................................................................................................................................550
4.20.5 Obtaining the Source Code....................................................................................................................................... 550
4.20.4
4.20.3 Planning the Paths for Software Porting..............................................................................................................549
4.20.2 Environment Requirements.......................................................................................................................................548
4.20.1 Introduction.................................................................................................................................................................... 547
4.20 pblat 2.1 Porting Guide (CentOS 7.6).......................................................................................................................547
4.19.8 Troubleshooting............................................................................................................................................................ 547
4.19.6.2 Creating Bowtie Index............................................................................................................................................. 546
4.19.6.1 Installing Bowtie........................................................................................................................................................544
4.19.6 4.19.1 Introduction.................................................................................................................................................................... 541
4.19 Bowtie 1.2.3 Porting Guide (CentOS 7.6)................................................................................................................ 541
4.26.4 Installing xsd............................................................................................................................................................... 585
4.26.4.5 Installing Amara........................................................................................................................................................ 584
4.26.4.3 Installing BOOST....................................................................................................................................................... 583
4.26.4.2 Installing PETSC......................................................................................................................................................... 582
4.26.4.1 Installing xerces and python-dateutil................................................................................................................ 581
4.26.4

4.26.1 Introduction.................................................................................................................................................................... 577
4.26 Chaste 2019.1 Porting Guide (CentOS 7.6)............................................................................................................ 577
4.25.7 Running and Verifying Abyss....................................................................................................................................575
4.25.6 Compiling and Installing Abyss................................................................................................................................575
4.25.5 Obtaining the Source Code....................................................................................................................................... 574
4.25.4

4.25.3 Paths for Software Porting........................................................................................................................................ 573
4.25.2 Environment Requirements.......................................................................................................................................572
4.25.1 Introduction.................................................................................................................................................................... 572
4.25 pysam 0.15.4 Porting Guide (CentOS 7.6)....................................................................................................................566
4.24.9 More Information......................................................................................................................................................... 569
4.24.8 Troubleshooting............................................................................................................................................................ 567
4.24.7 Running and Verifying Pysam.......................................................................................................................................570
4.24.6 Compiling and Installing Pysam..............................................................................................................................570
4.24.5 Obtaining the Source Code....................................................................................................................................... 570
4.24.4.2 Installing the Cython Package.............................................................................................................................. 569
4.24.4.1 Installing the Python Interpreter......................................................................................................................... 569
4.24.4

4.24.2 Environment Requirements.......................................................................................................................................567
4.24.1 Introduction.................................................................................................................................................................... 567
4.24.0.2 Installing lapack.........................................................................................................................................................564
4.24.0.1 Installing OpenBLAS.................................................................................................................................................564
4.24.0 Compiling and Installing XHMM..............................................................................................................................565
4.23.7 Running and Verifying XHMM....................................................................................................................................566
4.23.5 Obtaining the Source Code....................................................................................................................................... 566
4.23.4.2 Installing lapack.........................................................................................................................................................564
4.23.4.1 Installing OpenBLAS.................................................................................................................................................564
4.23.4

4.23.3 Planning the Paths for Software Porting..............................................................................................................562
4.23.2 Environment Requirements.......................................................................................................................................561
4.23.1 Introduction.................................................................................................................................................................... 561
4.23 XHMM Porting Guide (CentOS 7.6)..................................................................................................................................561
4.22.8 More Resources............................................................................................................................................................ 561
4.22.7 More Resources............................................................................................................................................................ 561
4.22.6 More Resources............................................................................................................................................................ 561
4.22.5 More Resources............................................................................................................................................................ 561
4.22.4 More Resources............................................................................................................................................................ 561
4.22.3 More Resources............................................................................................................................................................ 561
4.22.2 More Resources............................................................................................................................................................ 561
4.22.1 More Resources............................................................................................................................................................ 561
4.22 More Information.............................................................................................................................................................. 557
4.21 More Information.............................................................................................................................................................. 557
4.20 More Information.............................................................................................................................................................. 557
4.19 More Information.............................................................................................................................................................. 557
4.18 More Information.............................................................................................................................................................. 557
4.17 More Information.............................................................................................................................................................. 557
4.16 More Information.............................................................................................................................................................. 557
4.15 More Information.............................................................................................................................................................. 557
4.14 More Information.............................................................................................................................................................. 557
4.13 More Information.............................................................................................................................................................. 557
4.12 More Information.............................................................................................................................................................. 557
4.11 More Information.............................................................................................................................................................. 557
4.10 More Information.............................................................................................................................................................. 557
4.9 More Information.............................................................................................................................................................. 557
4.8 More Information.............................................................................................................................................................. 557
4.7 More Information.............................................................................................................................................................. 557
4.6 More Information.............................................................................................................................................................. 557
4.5 More Information.............................................................................................................................................................. 557
4.4 More Information.............................................................................................................................................................. 557
4.3 More Information.............................................................................................................................................................. 557
4.2 More Information.............................................................................................................................................................. 557
4.1 More Information.............................................................................................................................................................. 557
4.0 More Information.............................................................................................................................................................. 557

Copyright © Huawei Technologies Co., Ltd.

Issue 14 (2021-07-23)
4.35.1 Introduction .................................................................................................................................................................... 631
4.35.2 Environment Requirements ....................................................................................................................................... 630
4.35.3 Planning the Paths for Software Porting ...................................................................................................................... 631
4.35.4 Configuring the Compilation Environment ..................................................................................................................632
4.35.5 Obtaining the Source Code ......................................................................................................................................... 632
4.35.6 Compiling and Installing Minimap2.......................................................................................................................... 632
4.35.7 Running and Verifying Minimap2 ............................................................................................................................... 633
4.35.8 More Information .......................................................................................................................................................... 633

4.34.1 Introduction .................................................................................................................................................................... 626
4.34.2 Environment Requirements ....................................................................................................................................... 626
4.34.3 Planning the Paths for Software Porting ...................................................................................................................... 627
4.34.4 Configuring the Compilation Environment ..................................................................................................................628
4.34.5 Obtaining the Source Code ......................................................................................................................................... 629
4.34.6 Compiling and Installing BCFtools.......................................................................................................................... 629
4.34.7 Running and Verifying BCFtools ............................................................................................................................... 630
4.34.8 More Information .......................................................................................................................................................... 630

4.33.1 Introduction .................................................................................................................................................................... 617
4.33.2 Environment Requirements ....................................................................................................................................... 618
4.33.3 Planning the Paths for Software Porting ...................................................................................................................... 619
4.33.4 Configuring the Compilation Environment ..................................................................................................................620
4.33.5 Obtaining the Source Code ......................................................................................................................................... 620
4.33.6 Compiling and Installing BEAST 2 .......................................................................................................................... 620
4.33.7 Running and Verifying BEAST 2 ............................................................................................................................... 621
4.33.8 More Information .......................................................................................................................................................... 621

4.32.1 Introduction .................................................................................................................................................................... 614
4.32.2 Environment Requirements ....................................................................................................................................... 614
4.32.3 Planning the Paths for Software Porting ...................................................................................................................... 615
4.32.4 Configuring the Compilation Environment ..................................................................................................................616
4.32.5 Obtaining the Source Code ......................................................................................................................................... 616
4.32.6 Compiling and Installing BCFtools.......................................................................................................................... 616
4.32.7 Running and Verifying bcf tools ................................................................................................................................. 617
4.32.8 More Information .......................................................................................................................................................... 617

4.31.1 Introduction .................................................................................................................................................................... 610
4.31.2 Environment Requirements ....................................................................................................................................... 610
4.31.3 Planning the Path for Software Porting ......................................................................................................................... 611
4.31.4 Configuring the Compilation Environment ..................................................................................................................612
4.31.5 Obtaining the Source Code ......................................................................................................................................... 612
4.31.6 Compiling and Installing Minimap2 .......................................................................................................................... 612
4.31.7 Running and Verifying Minimap2 ............................................................................................................................... 613
4.31.8 More Information .......................................................................................................................................................... 613

4.30.1 Introduction .................................................................................................................................................................... 608
4.30.2 Environment Requirements ....................................................................................................................................... 608
4.30.3 Planning the Paths for Software Porting ...................................................................................................................... 609
4.30.4 Configuring the Compilation Environment ..................................................................................................................609
4.30.5 Obtaining the Source Code ......................................................................................................................................... 610
4.30.6 Compiling and Installing VarScan ............................................................................................................................... 610
4.30.7 Running and Verifying VarScan ................................................................................................................................. 611
4.30.8 More Information .......................................................................................................................................................... 611

4.29.1 Introduction .................................................................................................................................................................... 605
4.29.2 Environment Requirements ....................................................................................................................................... 605
4.29.3 Planning the Paths for Software Porting ...................................................................................................................... 606
4.29.4 Configuring the Compilation Environment ..................................................................................................................607
4.29.5 Obtaining the Source Code ......................................................................................................................................... 607
4.29.6 Compiling and Installing VarScan.......................................................................................................................... 607
4.29.7 Running and Verifying VarScan ................................................................................................................................. 608
4.29.8 More Information .......................................................................................................................................................... 608
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.35.6 Running and Installing Picard</td>
<td>632</td>
</tr>
<tr>
<td>4.35.7 Running and Verifying Picard</td>
<td>633</td>
</tr>
<tr>
<td>4.35.8 Troubleshooting</td>
<td>635</td>
</tr>
<tr>
<td>4.36 SOAPdenovo r241 Porting Guide (CentOS 7.6)</td>
<td>635</td>
</tr>
<tr>
<td>4.36.1 Introduction</td>
<td>635</td>
</tr>
<tr>
<td>4.36.2 Environment Requirements</td>
<td>636</td>
</tr>
<tr>
<td>4.36.3 Planning the Paths for Software Porting</td>
<td>636</td>
</tr>
<tr>
<td>4.36.4 Configuring the Compilation Environment</td>
<td>637</td>
</tr>
<tr>
<td>4.36.5 Obtaining the Source Code</td>
<td>637</td>
</tr>
<tr>
<td>4.36.6 Compiling and Installing SOAPdenovo</td>
<td>638</td>
</tr>
<tr>
<td>4.36.7 Running and Verifying SOAPdenovo</td>
<td>639</td>
</tr>
<tr>
<td>4.36.8 More Information</td>
<td>640</td>
</tr>
<tr>
<td>4.37 Blastz 2004-12-27 Porting Guide (CentOS 7.6)</td>
<td>640</td>
</tr>
<tr>
<td>4.37.1 Introduction</td>
<td>640</td>
</tr>
<tr>
<td>4.37.2 Environment Requirements</td>
<td>641</td>
</tr>
<tr>
<td>4.37.3 Paths for Software Porting</td>
<td>641</td>
</tr>
<tr>
<td>4.37.4 Configuring the Compilation Environment</td>
<td>642</td>
</tr>
<tr>
<td>4.37.5 Obtaining Source Code</td>
<td>642</td>
</tr>
<tr>
<td>4.37.6 Compilation and Installation</td>
<td>643</td>
</tr>
<tr>
<td>4.37.7 Running and Verifying Blastz</td>
<td>644</td>
</tr>
<tr>
<td>4.38 CAFE 5.0b2 Porting Guide (CentOS 7.6)</td>
<td>645</td>
</tr>
<tr>
<td>4.38.1 Introduction</td>
<td>645</td>
</tr>
<tr>
<td>4.38.2 Environment Requirements</td>
<td>645</td>
</tr>
<tr>
<td>4.38.3 Paths for Software Porting</td>
<td>646</td>
</tr>
<tr>
<td>4.38.4 Configuring the Compilation Environment</td>
<td>647</td>
</tr>
<tr>
<td>4.38.5 Obtaining Source Code</td>
<td>647</td>
</tr>
<tr>
<td>4.38.6 Compilation and Installation</td>
<td>647</td>
</tr>
<tr>
<td>4.38.7 Running and Verifying CAFE</td>
<td>648</td>
</tr>
<tr>
<td>4.38.8 More Resources</td>
<td>649</td>
</tr>
<tr>
<td>4.39 fastp Porting Guide (CentOS 7.6)</td>
<td>649</td>
</tr>
<tr>
<td>4.39.1 Introduction</td>
<td>649</td>
</tr>
<tr>
<td>4.39.2 Environment Requirements</td>
<td>649</td>
</tr>
<tr>
<td>4.39.3 Paths for Software Porting</td>
<td>650</td>
</tr>
<tr>
<td>4.39.4 Configuring the Compilation Environment</td>
<td>651</td>
</tr>
<tr>
<td>4.39.5 Obtaining Source Code</td>
<td>651</td>
</tr>
<tr>
<td>4.39.6 Compilation and Installation</td>
<td>651</td>
</tr>
<tr>
<td>4.39.7 Running and Verifying fastp</td>
<td>652</td>
</tr>
<tr>
<td>4.39.8 More Resources</td>
<td>652</td>
</tr>
<tr>
<td>4.40 GCE 1.0.2 Porting Guide (CentOS 7.6)</td>
<td>652</td>
</tr>
<tr>
<td>4.40.1 Introduction</td>
<td>652</td>
</tr>
<tr>
<td>4.40.2 Environment Requirements</td>
<td>653</td>
</tr>
<tr>
<td>4.40.3 Paths for Software Porting</td>
<td>654</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.45 megahit 1.2.9 Porting Guide (CentOS 7.6)</td>
<td>678</td>
</tr>
<tr>
<td>4.46.1 Introduction</td>
<td>678</td>
</tr>
<tr>
<td>4.46.2 Environment Requirements</td>
<td>679</td>
</tr>
<tr>
<td>4.46.3 Planning Paths for Software Porting</td>
<td>680</td>
</tr>
<tr>
<td>4.46.4 Configuring the Compilation Environment</td>
<td>680</td>
</tr>
<tr>
<td>4.46.4.1 Installing CMake</td>
<td>681</td>
</tr>
<tr>
<td>4.46.4.2 Installing URI</td>
<td>681</td>
</tr>
<tr>
<td>4.46.4.3 Installing Blat</td>
<td>682</td>
</tr>
<tr>
<td>4.46.4.4 Installing MariaDB</td>
<td>683</td>
</tr>
<tr>
<td>4.46.4.5 Installing SAMtools</td>
<td>683</td>
</tr>
<tr>
<td>4.46.5 Obtaining the Source Code</td>
<td>685</td>
</tr>
<tr>
<td>4.46.5.1 Basic Installation</td>
<td>685</td>
</tr>
<tr>
<td>4.46.5.2 (Optional) Installing MPI</td>
<td>686</td>
</tr>
<tr>
<td>4.46.5.3 (Optional) Installing MAFFT Extensions</td>
<td>686</td>
</tr>
<tr>
<td>4.46.6 Compiling and Installing PASA</td>
<td>688</td>
</tr>
<tr>
<td>4.46.7 Running and Verifying PASA</td>
<td>689</td>
</tr>
<tr>
<td>4.46.8 More Resources</td>
<td>690</td>
</tr>
<tr>
<td>4.47 Augustus 3.3.3 Porting Guide (CentOS 7.6)</td>
<td>692</td>
</tr>
<tr>
<td>4.47.1 Introduction</td>
<td>692</td>
</tr>
<tr>
<td>4.47.2 Environment Requirements</td>
<td>693</td>
</tr>
<tr>
<td>4.47.3 Planning the Paths for Software Porting</td>
<td>693</td>
</tr>
<tr>
<td>4.47.4 Configuring the Compilation Environment</td>
<td>694</td>
</tr>
<tr>
<td>4.47.4.1 Installing CMake</td>
<td>694</td>
</tr>
<tr>
<td>4.47.4.2 Installing HTSlib</td>
<td>695</td>
</tr>
<tr>
<td>4.47.4.3 Installing BCFtools</td>
<td>696</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.47.4.4 Installing SAMtools</td>
<td>697</td>
</tr>
<tr>
<td>4.47.4.5 Installing Bamtools</td>
<td>698</td>
</tr>
<tr>
<td>4.47.5 Obtaining the Source Code</td>
<td>698</td>
</tr>
<tr>
<td>4.47.6 Compiling and Installing AUGUSTUS</td>
<td>698</td>
</tr>
<tr>
<td>4.47.7 Running and Verifying AUGUSTUS</td>
<td>699</td>
</tr>
<tr>
<td>4.48 Delly 0.8.5 Porting Guide (CentOS 7.6)</td>
<td>700</td>
</tr>
<tr>
<td>4.48.1 Introduction</td>
<td>700</td>
</tr>
<tr>
<td>4.48.2 Environment Requirements</td>
<td>701</td>
</tr>
<tr>
<td>4.48.3 Paths for Software Porting</td>
<td>702</td>
</tr>
<tr>
<td>4.48.4 Configuring the Compilation Environment</td>
<td>702</td>
</tr>
<tr>
<td>4.48.4.1 Installing Boost</td>
<td>703</td>
</tr>
<tr>
<td>4.48.5 Obtaining Source Code</td>
<td>703</td>
</tr>
<tr>
<td>4.48.6 Compilation and Installation</td>
<td>704</td>
</tr>
<tr>
<td>4.48.7 Running and Verifying Delly</td>
<td>705</td>
</tr>
<tr>
<td>4.49 FastTree 2.1.11 Porting Guide (CentOS 7.6)</td>
<td>706</td>
</tr>
<tr>
<td>4.49.1 Introduction</td>
<td>707</td>
</tr>
<tr>
<td>4.49.2 Environment Requirements</td>
<td>707</td>
</tr>
<tr>
<td>4.49.3 Paths for Software Porting</td>
<td>708</td>
</tr>
<tr>
<td>4.49.4 Configuring the Compilation Environment</td>
<td>708</td>
</tr>
<tr>
<td>4.49.4.1 Installing GCC</td>
<td>709</td>
</tr>
<tr>
<td>4.49.5 Obtaining Source Code</td>
<td>709</td>
</tr>
<tr>
<td>4.49.6 Compilation and Installation</td>
<td>709</td>
</tr>
<tr>
<td>4.49.7 Running and Verifying FastTree</td>
<td>710</td>
</tr>
<tr>
<td>4.49.8 More Resources</td>
<td>710</td>
</tr>
<tr>
<td>4.50 Jellyfish 2.2.10 Porting Guide (Centos 7.6)</td>
<td>710</td>
</tr>
<tr>
<td>4.50.1 Introduction</td>
<td>710</td>
</tr>
<tr>
<td>4.50.2 Environment Requirements</td>
<td>711</td>
</tr>
<tr>
<td>4.50.3 Planning Paths for Software Porting</td>
<td>712</td>
</tr>
<tr>
<td>4.50.4 Configuring the Compilation Environment</td>
<td>712</td>
</tr>
<tr>
<td>4.50.5 Obtaining the Source Code</td>
<td>713</td>
</tr>
<tr>
<td>4.50.6 Compiling and Installing Jellyfish</td>
<td>713</td>
</tr>
<tr>
<td>4.50.6.1 Installing Jellyfish</td>
<td>713</td>
</tr>
<tr>
<td>4.50.7 Running and Verifying Jellyfish</td>
<td>714</td>
</tr>
<tr>
<td>4.51 PLINK 1.9 Porting Guide (CentOS 7.6)</td>
<td>714</td>
</tr>
<tr>
<td>4.51.1 Introduction</td>
<td>714</td>
</tr>
<tr>
<td>4.51.2 Environment Requirements</td>
<td>714</td>
</tr>
<tr>
<td>4.51.3 Planning Paths for Software Porting</td>
<td>714</td>
</tr>
<tr>
<td>4.51.4 Obtaining the Source Code</td>
<td>715</td>
</tr>
<tr>
<td>4.51.5 Configuring the Compilation Environment</td>
<td>716</td>
</tr>
<tr>
<td>4.51.6 Compiling and Installing PLINK</td>
<td>716</td>
</tr>
<tr>
<td>4.51.7 More Resources</td>
<td>717</td>
</tr>
<tr>
<td>4.52 SNAP 2013-11-29 Porting Guide (CentOS 7.6)</td>
<td>717</td>
</tr>
</tbody>
</table>
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.52.1</td>
<td>Introduction</td>
<td>717</td>
</tr>
<tr>
<td>4.52.2</td>
<td>Environment Requirements</td>
<td>718</td>
</tr>
<tr>
<td>4.52.3</td>
<td>Paths for Software Porting</td>
<td>719</td>
</tr>
<tr>
<td>4.52.4</td>
<td>Configuring the Compilation Environment</td>
<td>719</td>
</tr>
<tr>
<td>4.52.4.1</td>
<td>Installing MAKER</td>
<td>720</td>
</tr>
<tr>
<td>4.52.5</td>
<td>Obtaining the Source Code</td>
<td>720</td>
</tr>
<tr>
<td>4.52.6</td>
<td>Compiling and Installing SNAP</td>
<td>720</td>
</tr>
<tr>
<td>4.52.6.1</td>
<td>Installing SNAP</td>
<td>720</td>
</tr>
<tr>
<td>4.52.7</td>
<td>Running and Verifying SNAP</td>
<td>722</td>
</tr>
<tr>
<td>4.52.8</td>
<td>More Resources</td>
<td>723</td>
</tr>
<tr>
<td>4.53</td>
<td>kmersGWAS 0.2 Porting Guide (CentOS 7.6)</td>
<td>723</td>
</tr>
<tr>
<td>4.53.1</td>
<td>Introduction</td>
<td>724</td>
</tr>
<tr>
<td>4.53.2</td>
<td>Environment Requirements</td>
<td>724</td>
</tr>
<tr>
<td>4.53.3</td>
<td>Planning the Paths for Software Porting</td>
<td>725</td>
</tr>
<tr>
<td>4.53.4</td>
<td>Configuring the Compilation Environment</td>
<td>726</td>
</tr>
<tr>
<td>4.53.4.1</td>
<td>Installing R</td>
<td>727</td>
</tr>
<tr>
<td>4.53.4.2</td>
<td>Installing CMake</td>
<td>727</td>
</tr>
<tr>
<td>4.53.4.3</td>
<td>Installing OpenBLAS</td>
<td>727</td>
</tr>
<tr>
<td>4.53.4.4</td>
<td>Installing Eigen</td>
<td>728</td>
</tr>
<tr>
<td>4.53.4.5</td>
<td>Installing GSL</td>
<td>729</td>
</tr>
<tr>
<td>4.53.4.6</td>
<td>Installing Gemma</td>
<td>729</td>
</tr>
<tr>
<td>4.53.5</td>
<td>Obtaining the Source Code</td>
<td>730</td>
</tr>
<tr>
<td>4.53.6</td>
<td>Compiling and Installing kmersGWAS</td>
<td>731</td>
</tr>
<tr>
<td>4.53.7</td>
<td>Running and Verifying kmersGWAS</td>
<td>732</td>
</tr>
<tr>
<td>4.54</td>
<td>PAML 4.9j Porting Guide (CentOS 7.6)</td>
<td>733</td>
</tr>
<tr>
<td>4.54.1</td>
<td>Introduction</td>
<td>733</td>
</tr>
<tr>
<td>4.54.2</td>
<td>Environment Requirements</td>
<td>733</td>
</tr>
<tr>
<td>4.54.3</td>
<td>Planning Paths for Software Porting</td>
<td>734</td>
</tr>
<tr>
<td>4.54.4</td>
<td>Obtaining the Source Code</td>
<td>735</td>
</tr>
<tr>
<td>4.54.5</td>
<td>Compiling and Installing PAML</td>
<td>735</td>
</tr>
<tr>
<td>4.54.6</td>
<td>Running and Verifying PAML</td>
<td>736</td>
</tr>
<tr>
<td>4.54.7</td>
<td>More Resources</td>
<td>736</td>
</tr>
<tr>
<td>4.55</td>
<td>Diamond 2.0.4 Porting Guide (CentOS 7.6)</td>
<td>737</td>
</tr>
<tr>
<td>4.55.1</td>
<td>Introduction</td>
<td>737</td>
</tr>
<tr>
<td>4.55.2</td>
<td>Environment Requirements</td>
<td>737</td>
</tr>
<tr>
<td>4.55.3</td>
<td>Paths for Software Porting</td>
<td>738</td>
</tr>
<tr>
<td>4.55.4</td>
<td>Configuring the Compilation Environment</td>
<td>739</td>
</tr>
<tr>
<td>4.55.4.1</td>
<td>Installing CMake</td>
<td>739</td>
</tr>
<tr>
<td>4.55.5</td>
<td>Obtaining Source Code</td>
<td>740</td>
</tr>
<tr>
<td>4.55.6</td>
<td>Compilation and Installation</td>
<td>740</td>
</tr>
<tr>
<td>4.55.7</td>
<td>Running and Verifying Diamond</td>
<td>741</td>
</tr>
<tr>
<td>4.55.8</td>
<td>More Resources</td>
<td>741</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>4.56 HISAT 0.1.6 Porting Guide (CentOS 7.6)</td>
<td>741</td>
<td></td>
</tr>
<tr>
<td>4.56.1 Introduction</td>
<td>741</td>
<td></td>
</tr>
<tr>
<td>4.56.2 Environment Requirements</td>
<td>742</td>
<td></td>
</tr>
<tr>
<td>4.56.3 Paths for Software Porting</td>
<td>743</td>
<td></td>
</tr>
<tr>
<td>4.56.4 Configuring the Compilation Environment</td>
<td>743</td>
<td></td>
</tr>
<tr>
<td>4.56.5 Obtaining Source Code</td>
<td>744</td>
<td></td>
</tr>
<tr>
<td>4.56.6 Compilation and Installation</td>
<td>744</td>
<td></td>
</tr>
<tr>
<td>4.56.7 Running and Verifying HISAT</td>
<td>747</td>
<td></td>
</tr>
<tr>
<td>4.57 MACS2 2.1.4 Porting Guide (CentOS 7.6)</td>
<td>747</td>
<td></td>
</tr>
<tr>
<td>4.57.1 Introduction</td>
<td>747</td>
<td></td>
</tr>
<tr>
<td>4.57.2 Environment Requirements</td>
<td>748</td>
<td></td>
</tr>
<tr>
<td>4.57.3 Planning Paths for Software Porting</td>
<td>749</td>
<td></td>
</tr>
<tr>
<td>4.57.4 Configuring the Compilation Environment</td>
<td>749</td>
<td></td>
</tr>
<tr>
<td>4.57.4.1 Installing python-devel</td>
<td>750</td>
<td></td>
</tr>
<tr>
<td>4.57.4.2 Installing Cython</td>
<td>750</td>
<td></td>
</tr>
<tr>
<td>4.57.4.3 Installing NumPy</td>
<td>751</td>
<td></td>
</tr>
<tr>
<td>4.57.5 Obtaining the Source Code</td>
<td>751</td>
<td></td>
</tr>
<tr>
<td>4.57.6 Compiling and Installing MACS2</td>
<td>751</td>
<td></td>
</tr>
<tr>
<td>4.57.7 Running and Verifying MACS2</td>
<td>752</td>
<td></td>
</tr>
<tr>
<td>4.57.8 More Resources</td>
<td>754</td>
<td></td>
</tr>
<tr>
<td>4.58 Trinity 2.11.0 Porting Guide (CentOS 7.6)</td>
<td>754</td>
<td></td>
</tr>
<tr>
<td>4.58.1 Introduction</td>
<td>754</td>
<td></td>
</tr>
<tr>
<td>4.58.2 Environment Requirements</td>
<td>755</td>
<td></td>
</tr>
<tr>
<td>4.58.3 Planning the Paths for Software Porting</td>
<td>757</td>
<td></td>
</tr>
<tr>
<td>4.58.4 Configuring the Compilation Environment</td>
<td>757</td>
<td></td>
</tr>
<tr>
<td>4.58.4.1 Installing All Dependencies Using the yum Command</td>
<td>758</td>
<td></td>
</tr>
<tr>
<td>4.58.4.2 Installing Boost</td>
<td>758</td>
<td></td>
</tr>
<tr>
<td>4.58.4.3 Installing CMake</td>
<td>759</td>
<td></td>
</tr>
<tr>
<td>4.58.4.4 Installing Salmon</td>
<td>759</td>
<td></td>
</tr>
<tr>
<td>4.58.4.5 Installing HTSlib</td>
<td>760</td>
<td></td>
</tr>
<tr>
<td>4.58.4.6 Installing SAMtools</td>
<td>761</td>
<td></td>
</tr>
<tr>
<td>4.58.4.7 Installing Jellyfish</td>
<td>761</td>
<td></td>
</tr>
<tr>
<td>4.58.4.8 Installing Bowtie 2</td>
<td>762</td>
<td></td>
</tr>
<tr>
<td>4.58.4.9 Installing Python</td>
<td>762</td>
<td></td>
</tr>
<tr>
<td>4.58.4.10 Installing Cython</td>
<td>763</td>
<td></td>
</tr>
<tr>
<td>4.58.4.11 Installing NumPy</td>
<td>763</td>
<td></td>
</tr>
<tr>
<td>4.58.5 Obtaining the Source Code</td>
<td>764</td>
<td></td>
</tr>
<tr>
<td>4.58.6 Compiling and Installing Trinity</td>
<td>764</td>
<td></td>
</tr>
<tr>
<td>4.58.7 Running and Verifying Trinity</td>
<td>766</td>
<td></td>
</tr>
<tr>
<td>4.59 LDBlockshow 1.36 Porting Guide (CentOS 7.6)</td>
<td>767</td>
<td></td>
</tr>
<tr>
<td>4.59.1 Introduction</td>
<td>767</td>
<td></td>
</tr>
<tr>
<td>4.59.2 Environment Requirements</td>
<td>767</td>
<td></td>
</tr>
</tbody>
</table>
4.59.3 Planning Paths for Software Porting ................................................................. 768
4.59.4 Obtaining the Source Code .............................................................................. 769
4.59.5 Configuring the Compilation Environment .................................................... 769
4.59.6 Compiling and Installing LDBlockShow ....................................................... 770
4.59.7 Running and Verifying LDBlockShow ............................................................. 771

5 Change History ............................................................................................................. 772
1 Meteorology

1.1 WRF 3.8.1 Porting Guide (CentOS 7.6)
1.2 CAMx 6.50 Porting Guide (CentOS 7.6)
1.3 CESM 2.1.1 Porting Guide (CentOS 7.6)
1.4 NEMO V3.6 Porting Guide (CentOS 7.6)
1.5 ROMS 3.6 Porting Guide (CentOS 7.6)
1.6 WAVEWATCH III 6.07.1 Porting Guide (CentOS 7.6)
1.7 Bifrost V1.1.1 Porting Guide (CentOS 7.6)
1.8 FVCOM 4.1 Porting Guide (CentOS 7.6)
1.9 MITGCM c67o Porting Guide (CentOS 7.6)
1.10 POM POM2K Porting Guide (CentOS 7.6)
1.11 MOM 6 Porting Guide (CentOS 7.6)
1.12 SMOKE 4.7 Porting Guide (CentOS 7.6)
1.13 SPECFEM3D GLOBE 7.0.0 Porting Guide (CentOS 7.6)
1.14 CMAQ 5.3.1 Porting Guide (CentOS 7.6)
1.15 SWAN 41.31 Porting Guide (CentOS 7.6)
1.16 CDO 1.9.8 Porting Guide (CentOS 7.6)
1.17 ecCodes 2.18.0 Porting Guide (CentOS 7.6)
1.18 ecFlow 5.5.2 Porting Guide (CentOS 7.6)
1.19 NCEPLIBS 1.2.0 Porting Guide (CentOS 7.6)
1.20 Pyferret 7.6.0 Porting Guide (CentOS 7.6)
1.21 ncl 6.3.0 Porting Guide (CentOS 7.6)
1.1 WRF 3.8.1 Porting Guide (CentOS 7.6)

1.1.1 Introduction

The Weather Research and Forecasting (WRF) Model can be used for fine-scale weather simulation and forecasting, which is one of the important application scenarios of high-performance computing (HPC).

For more information about the WRF, visit the official WRF website.

Programming language: C/Fortran

Brief description: mesoscale weather forecasting model

Open-source protocol: public domain

Recommended Version

The recommended version is WRF V3.8.1.

1.1.2 Environment Requirements

Hardware Requirements

Table 1-1 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table Software requirements lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDF5</td>
<td>1.10.1</td>
</tr>
<tr>
<td>PNETCDF</td>
<td>1.9.0</td>
</tr>
<tr>
<td>NETCDF</td>
<td>4.4.1.1</td>
</tr>
</tbody>
</table>

Download Address

- HDF5: https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/
- NETCDF: https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1
### OS Requirements

*Table 1-3* lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 1.1.3 Planning the Paths for Software Porting

*Table Paths for software porting* lists the software installation paths involved in the WRF software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning Data for Installation in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/PNETCDF</td>
<td>Installation path of PNETCDF 1.9.0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NETCDF 4.4.1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NETCDF-Fortran 4.4.1</td>
<td></td>
</tr>
</tbody>
</table>
1.1.4 Configuring theCompilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-5 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Cluster Environment&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>Installing HDF5</td>
<td>For details, see <a href="#">1.1.4.2 Installing HDF5</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Installing PNETCDF</td>
<td>For details, see <a href="#">1.1.4.3 Installing PNETCDF</a>.</td>
</tr>
<tr>
<td>4</td>
<td>Installing NETCDF-C</td>
<td>For details, see <a href="#">1.1.4.4 Installing NETCDF-C</a>.</td>
</tr>
<tr>
<td>5</td>
<td>Installing NETCDF-FORTRAN</td>
<td>For details, see <a href="#">1.1.4.5 Installing NETCDF-FORTRAN</a>.</td>
</tr>
<tr>
<td>6</td>
<td>Installing OPTIMIZED-ROUTINES</td>
<td>For details, see <a href="#">1.1.4.6 Installing OPTIMIZED-ROUTINES</a>.</td>
</tr>
<tr>
<td>7</td>
<td>Installing ClusterShell</td>
<td>For details, see <a href="#">1.1.4.7 Installing ClusterShell</a>.</td>
</tr>
</tbody>
</table>
1.1.4.1 Setting Up the Basic Environment

For details, see "Setting Up the Environment for the Cluster Scenario" in HPC Solution Basic Environment Setup Guide.

1.1.4.2 Installing HDF5

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to install HDF5.

```bash
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
.tar -xvf hdf5-1.10.1.tar.gz
.cd hdf5-1.10.1
.mkdir -p /path/to/HDF5
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
.make -j 16
.make install
```

Step 3  Run the following commands to check whether HDF5 is successfully installed:

```bash
cd /path/to/HDF5
.ls lib
```

If the following information is displayed, the installation is successful.

```
----End
```

1.1.4.3 Installing PNETCDF

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to install PNETCDF.

```bash
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
.tar -xvf parallel-netcdf-1.9.0.tar.bz2
```
cd parallel-netcdf-1.9.0
mkdir -p /path/to/PNETCDF
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu
CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC"
CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
make -j 16
make install

NOTE
If the environment variables of the compiler and MPI have been declared, skip the first two
commands for declaring environment variables in this step.

Step 3  Run the following commands to check whether PNETCDF is successfully installed.

cd /path/to/PNETCDF
ls lib

If the following information is displayed, the installation is successful.

----End

1.1.4.4 Installing NETCDF-C

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to install NETCDF-C.

echo "export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:

tar -xvf netcdf-4.4.1.1.tar.gz

cd netcdf-c-4.4.1.1

mkdir -p /path/to/NETCDF

./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu
enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static
--enable-pnetcdf --enable-largefile CC=mpicc CXX=mpicxx
FC=mpifort F77=mpifort

CCFLAGS="-I/path/to/HDF5/include -I/path/to/PNETCDF/include"
LDFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/PNETCDF/lib"
CPPFLAGS="-I/path/to/HDF5/include -I/path/to/PNETCDF/include"

make -j 16
1.1.4.5 Installing NETCDF-FORTRAN

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install NETCDF-FORTRAN.

```bash
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:

http://www.example.com

$LD_LIBRARY_PATH

tar -xvf netcdf-fortran-4.4.1.tar.gz

cd netcdf-fortran-4.4.1

./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static

CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include" LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib"

CFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include" CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"

 FCFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"

make -j 16

make install

NOTE

If the environment variables of the compiler and MPI have been declared, skip the first two commands for declaring environment variables in this step.

Step 3 Check whether NETCDF-C NETCDF-FORTRAN is successfully installed.

```bash
cd /path/to/NETCDF

ls lib

```

If the following information is displayed, the installation is successful.

```
libnetcdf.a libnetcdf.so.1 libnetcdf.so.1.6 libnetcdf.so.1.6.0 libnetcdf.so.1.6.0.2
libnetcdf.a libnetcdf.so.1 libnetcdf.so.1.6 libnetcdf.so.1.6.0 libnetcdf.so.1.6.0.2

```
1.1.4.6 Installing OPTIMIZED-ROUTINES

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to install OPTIMIZED-ROUTINES.

```bash
yum install glibc* glibm* -y

tar -xvf optimized-routines-201910.tar.gz

cd optimized-routines-201910

mkdir -p /path/to/MATH

cp config.mk.dist config.mk

make -j 16

cp -r ./build/* /path/to/MATH
```

**Step 3** Check whether the installation is successful.

- `cd /path/to/MATH`
- `ls lib`

If the following information is displayed, the installation is successful.

```
[root@z288node24 MATH]# cd /path/to/MATH/
[root@z288node24 MATH]# ls lib/
libmathlib.a libmathlib.so libstringlib.a libstringlib.so
```

----End

1.1.4.7 Installing ClusterShell

**Step 1** Run the following commands to download the ClusterShell RPM installation package `clustershell-1.8.2-1.el7.noarch.rpm` and its dependency `python2-clustershell-1.8.2-1.el7.noarch.rpm`:

```bash
wget https://mirrors.tuna.tsinghua.edu.cn/epel/7/aarch64/Packages/c/
clustershell-1.8.2-1.el7.noarch.rpm

wget https://mirrors.tuna.tsinghua.edu.cn/epel/7/aarch64/Packages/p/
python2-clustershell-1.8.2-1.el7.noarch.rpm
```

**Step 2** Run the following commands to install ClusterShell:

```bash
yum install python-setuptools –y

rpm -ivh python2-clustershell-1.8.2-1.el7.noarch.rpm
clustershell-1.8.2-1.el7.noarch.rpm
```

**NOTE**

ClusterShell is installed to clear the cache of each node when multiple nodes are running. If only one node is running, you do not need to install ClusterShell.

----End
1.1.5 Obtaining Source Code

**Step 1** Download the WRF source code package **WRF-3.8.1.tar.gz**.

Download address: [https://github.com/wrf-model/WRF/releases/tag/V3.8.1](https://github.com/wrf-model/WRF/releases/tag/V3.8.1)

**Step 2** Use the SFTP tool to copy the WRF source code package to the `/path/to/WRF` directory on the server.

----End

1.1.6 Compiling and Installing WRF

**Step 1** Use PuTTY to log in to the server as the `root` user.

**Step 2** Decompress the WRF installation package.

```
cd /path/to/WRF

    tar -xvf WRF-3.8.1.tar.gz
```

**Step 3** Run the following command to switch to WRF source code directory:

```
cd WRF-3.8.1
```

**Step 4** Edit the `arch/configure_new.defaults` file and add the following information before line 1856:

```bash
#ARCH   Linux   aarch64,gnu OpenMPI #serial smpar dmpar dm+smp
DESCRIPTION     =       GNU ($SFC/$SCC)
DMPARALLEL      =        1
OMPCPP          =        -D_OPENOMP
OMP             =        -fopenmp
OMPCC           =        -fopenmp
SFC             =       gfortran
SCC             =       gcc
CCOMP           =       gcc
DM_FC           =       mpif90 -f90=$(SFC)
DM_CC           =       mpicc -cc=$(SCC) -DMPI2_SUPPORT
FC              =       CONFIGURE_FC
CC              =       CONFIGURE_CC
LD              =       $(FC)
RWORDSIZE       =       CONFIGURE_RWORDSIZE
PROMOTION       =        #-fddefault-real-8
ARCH_LOCAL      =       -DNONSTANDARD_SYSTEM_SUBR -DWRF_USE_CLM
CFLAGS_LOCAL    =       -w -O3 -c -march=armv8.2-a -L/path/to/MATH/lib -lmathlib
LDFLAGS_LOCAL   =       
CPLUSPLUSLIB    =       
ESMF_LDFLAG     =      $(CPLUSPLUSLIB)
FCOPTIM         =       -O3 -ftree-vectorize -funroll-loops -march=armv8.2-a -L/path/to/MATH/lib -lmathlib
FCREDUCEDOPT    =       $(FCOPTIM)
FCNOOPT         =       -O0
FCDEBUG         =       # -g $(FCNOOPT) # -fbacktrace -ggdb -fcheck=bounds,do,mem,pointer -ffpe-trap=invalid,zero,overflow
FORMAT_FIXED    =       -ffixed-form
FORMAT_FREE     =       -ffree-form -ffree-line-length-none
FCPREFIX        =       
BYTESWAPIO      =       -fconvert=big-endian -frecord-marker=4
FCBASEOPTS_NO_G =       -w $(FORMAT_FREE) $(BYTESWAPIO)
FCBASEOPTS      =       $(FCBASEOPTS_NO_G) $(FCDEBUG)
MODULE_SRCH_FLAG =       
TRADFLAG        =       -traditional
CPP             =       /lib/cpp -P
AR              =       ar
ARFLAGS         =       ru
```
Step 5  Run the following commands to configure the pre-compilation environment:

```bash
echo WRFIO_NCD_LARGE_FILE_SUPPORT=1
```

```bash
echo NETCDF=/path/to/NETCDF
```

```bash
echo HDF5=/path/to/HDF5
```

```bash
echo PNETCDF=/path/to/PNETCDF
```

```bash
echo CPPFLAGS="-I$HDF5/include -I$PNETCDF/include -I$NETCDF/include"
```

```bash
echo LDFLAGS="-L$HDF5/lib -L$PNETCDF/lib -L$NETCDF/lib -lnetcdf -lnetcdff -lpnetcdf -lhdf5_hl -lhdf5 -lz"
```

Step 6  Run the following command to generate a configuration file:

```bash
echo 4 | ./configure
```

Step 7  Run the following command to modify line 3125 of the ./phys/module_cu_g3.F file:

```bash
echo 4 | ./configure
```

```bash
echo HDF5=/path/to/NETCDF
```

```bash
echo HDF5=/path/to/HDF5
```

```bash
echo PNETCDF=/path/to/PNETCDF
```

```bash
echo CPPFLAGS="-I$HDF5/include -I$PNETCDF/include -I$NETCDF/include"
```

```bash
echo LDFLAGS="-L$HDF5/lib -L$PNETCDF/lib -L$NETCDF/lib -lnetcdf -lnetcdff -lpnetcdf -lhdf5_hl -lhdf5 -lz"
```

Step 8  Run the following commands to load the compiler and set the MPI environment variables.

```bash
echo PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
```

```bash
echo LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
```

```bash
echo $LD_LIBRARY_PATH
```

Step 9  Run the following command to perform compilation and installation:

```bash
./compile -j 16 em_real 2>&1 | tee -a compile.log
```

Step 10  Check whether the installation is successful.

```bash
ls main
```

If the following information is displayed (the wrf.exe file is generated), the installation is successful. The installation of the main program takes approximately 10 minutes.

```
-----End
```
1.1.7 Running and Verifying WRF

Step 1  Obtain the case file from the following link:

https://www2.mmm.ucar.edu/WG2bench/conus_2.5_v3/1-RST/

Three target files: wrfrst_d01_2005-06-04_06_00_00, namelist.input, and wrfbdy_d01

Step 2  Run the following commands on PuTTY to create the /path/to/WRFTEST directory and upload the case to the directory:

```
mkdir -p /path/to/WRFTEST
```

```
cd /path/to/WRFTEST
```

```
wget https://www2.mmm.ucar.edu/WG2bench/conus_2.5_v3/1-RST/RST/ -r -c -np -nH --cut-dirs 2 --restrict-file-names=nocontrol -e robots=off --reject "index.html"
```

```
wget https://www2.mmm.ucar.edu/WG2bench/conus_2.5_v3/wrfbdy_d01.gz -r -c -np -nH --cut-dirs 2 --restrict-file-names=nocontrol -e robots=off --reject "index.html"
```

```
wget https://www2.mmm.ucar.edu/WG2bench/conus_2.5_v3/1-RST/namelist.input -r -c -np -nH --cut-dirs 2 --restrict-file-names=nocontrol -e robots=off --reject "index.html"
```

Step 3  Run the following command to place namelist.input in the test directory and delete unnecessary directories:

```
mv 1-RST/namelist.input ./
```

Step 4  Run the following commands to generate a case and delete unnecessary directories:

```
cat 1-RST/RST/rst_6hr* | gunzip -c > wrfrst_d01_2005-06-04_06_00_00
rm -rf 1-RST
```

Step 5  Run the following command to decompress the wrfbdy_d01 file:

```
gunzip wrfbdy_d01.gz
```

Step 6  Run the following command to modify the namelist.input file:

```
 vim namelist.input
```

```
&time_control
  run_days = 0,
  run_hours = 3,
  run_minutes = 0,
  run_seconds = 0,
  start_year = 2005,
  start_month = 06,
  start_day = 04,
  start_hour = 06,
  start_minute = 00,
  start_second = 00,
  end_year = 2005,
  end_month = 06,
  end_day = 04,
  end_hour = 12,
```
end_minute = 00,
end_second = 00,
**interval_seconds** = 10800
input_from_file = .true.,
history_interval = 180,
frames_per_outfile = 1,
no_cols = .true.,
restart = .true.,
restart_interval = 360,
io_form_history = 11,
io_form_restart = 11,
io_form_input = 2,
io_form_boundary = 2,
io_form_auxhist2 = 2,
debug_level = 0
/
&domains
time_step = 15,
time_step_fract_num = 0,
time_step_fract_den = 1,
max_dom = 1,
s_we = 1,
e_we = 1501,
s_sn = 1,
e_sn = 1201,
s_vert = 1,
e_vert = 35,
dx = 2500,
dy = 2500,
grid_id = 1,
p_parent_id = 0,
i_parent_start = 0,
j_parent_start = 0,
p_parent_grid_ratio = 1,
p_parent_time_step_ratio = 1,
feedback = 1,
smooth_option = 0,
lagrange_order = 2,
interp_type = 2,
extrap_type = 2,
t_extrap_type = 2,
use_surface = .true.
use_levels_below_ground = .true.
num_metgrid_levels = 40,
eta_levels = (1.00000, 0.99258, 0.98275, 0.96996, 0.95372,
0.93557, 0.90913, 0.87957, 0.84531, 0.80683, 0.76467,
0.71940, 0.67163, 0.62198, 0.57108, 0.51956, 0.46803,
0.42030, 0.37613, 0.33532, 0.29764, 0.26290, 0.23092,
0.20152, 0.17452, 0.14978, 0.12714, 0.10646, 0.08761,
0.07045, 0.05466, 0.03981, 0.02580, 0.01258, 0.00000)
/
&physics
mp_physics = 4,
mp_zero_out = 2,
mp_zero_out_thresh = 1.e-9,
ra_lw_physics = 1,
ra_sw_physics = 1,
radiat = 10,
sf_sfclay_physics = 1,
sf_surface_physics = 2,
bl_pbl_physics = 1,
bldt = 0,
cu_physics = 0,
cudt = 5,
isflx = 1,
isn = 0,
icloud = 1,
surface_input_source = 1,
| num_soil_layers | = 4, |
| maxiens        | = 1, |
| maxens         | = 1, |
| maxens2        | = 1, |
| maxens3        | = 1, |
| ensdim         | = 1, |
| \&dynamics     | w_damping = 1, |
| diff_opt       | = 1, |
| km_opt         | = 4, |
| khdif          | = 0, |
| kvdif          | = 0, |
| non_hydrostatic|= .true., |
| use_baseparam_fr_nml|= .t., |
| \&bdy_control  | spec_bdy_width = 5, |
| spec_zone      | = 1, |
| relax_zone     | = 4, |
| specified      | = .true., |
| nested         | = .false., |
| \&namelist_quilt| nio_tasks_per_group = 0, |
|                | nio_groups = 1, |

**Step 7** Run the following command to copy all files in the `/path/to/WRF/WRF-3.8.1/run` directory to the `/path/to/WRFTEST` directory:

```bash
cp /path/to/WRF/WRF-3.8.1/run/* /path/to/WRFTEST/
```

**Step 8** Run the following commands to load environment variables:

```bash
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:SPATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:/path/to/NETCDF/lib:/path/to/PNETCDF/lib:/path/to/HDF5/lib:/path/to/MATH/lib:
SLD_LIBRARY_PATH
```

**Step 9** Run the following command to create the `hostfile` file:

1. `vi hostfile`
2. Press `i` to go to the edit mode.
   
   ```
   Node1
   Node2
   ```
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

**Step 10** Run the following command to start the WRF program.

```bash
clush --hostfile hostfile "sync;echo 3 > /proc/sys/vm/drop_caches"
```

- If an IB network is used, run the following command:
  ```bash
time -p `which mpirun` --allow-run-as-root -x PATH=$PATH -x LD_LIBRARY_PATH=SLD_LIBRARY_PATH -x LD_PRELOAD=/path/to/MATH/lib/libmathlib.so -x OMP_NUM_THREADS=4 -map-by ppr:32:node:pe=4 --bind-to core --hostfile hostfile --mca pml ucx ./wrf.exe
```

- If an RoCE network is used, run the following command:
  ```bash
time -p mpirun --allow-run-as-root -hostfile hostfile -x PATH -x LD_LIBRARY_PATH -x LD_PRELOAD=/path/to/MATH/lib/libmathlib.so -x OMP_NUM_THREADS=4 -map-by ppr:24:node:pe=4 -bind-to core -display-
map -mca pml ucx -mca btl ^vader,tcp,openib,uct -x UCX_TLS=self,sm,rc -x UCX_NET_DEVICES=mlx5_0:1 -x UCX_IB_GID_INDEX=5 ./wrf.exe

- **--hostfile** hostfile: specifies the list of node names to be used.
- **-x OMP_NUM_THREADS=4**: specifies the number of threads to be used.
- **-map-by ppr:32:node:pe=4**: specifies how processes and threads are bound. Each node has 32 processes, and each process has four threads. (The 128-core CPU on a single node is used as an example.)
- **UCX_NET_DEVICES**: specifies the RoCE network port.
- **UCX_IB_GID_INDEX**: specifies the RoCE network type.

**Step 11** Verify that the program ends normally.

```
less rsl.out.0000
```

---End

## 1.2 CAMx 6.50 Porting Guide (CentOS 7.6)

### 1.2.1 Introduction

CAMx is a multi-scale photochemical modeling system for gas and particulate air pollution.

For more information about CAMx, visit the [official CAMx website](#).

Language: Fortran

Brief description: multi-scale photochemical modeling system

Open-source protocol: user-defined open-source protocol

### Recommended Version

The recommended version is CAMx 6.50.

### 1.2.2 Environment Requirements

#### Hardware Requirements

[Table 1-6](#) lists the hardware requirements.
### Table 1-6 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

### Software Requirements

Table 1-7 lists the software requirements.

#### Table 1-7 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute instance file 1</td>
<td>CAMx6-x-test_run-inputs_met-140402.tgz</td>
<td><a href="http://www.camx.com/download/camx-test-case.aspx">http://www.camx.com/download/camx-test-case.aspx</a></td>
</tr>
</tbody>
</table>

### OS Requirements

Table 1-8 lists the OS requirements.

#### Table 1-8 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 1.2.3 Planning the Paths for Software Porting

Table Paths for software porting lists the software installation paths involved in the CAMx software porting.
Table 1-9 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>environment</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>/path/to/CAMX</td>
<td>Installation path of CAMx</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>paths used in the commands in this document are examples only. Use the actual paths planned during the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/CASE</td>
<td>Path for storing the CAMx test cases</td>
<td></td>
</tr>
</tbody>
</table>

1.2.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.

**Procedure**

Table 1-10 Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

1.2.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the CAMx_v6-50-src-180430.tgz installation package.

Download address: [http://www.camx.com/download/default.aspx](http://www.camx.com/download/default.aspx)

**Step 2** Use SFTP to upload the CAMx installation package to the /path/to/CAMX directory on the server.

----End
1.2.6 Compiling and Installing CAMx

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the installation file package:

```
mkdir -p /path/to/CAMX
cd /path/to/CAMX
mkdir -p CAMX_v6-50
tar xzvf CAMx_v6-50-src-180430.tgz -C CAMX_v6-50
```

**Step 3** Run the following command to go to source code directory:

```
cd CAMX_v6-50
```

**Step 4** Run the following command to modify the Makefile file:

1. `vi Makefile`
   2. Press I to enter the insert mode and modify line 64 in the Makefile file. Pay attention to the information in bold.
      
      ```
      MPI_INST = /path/to/OPENMPI
      # lib = -L$(MPI_INST)/lib -lmpi
      # CFLAGS = -mmodel=small -O2 -l $(INC)
      ```
      3. Press Esc, enter `.wq!`, and press Enter to save the file and exit.

**Step 5** Run the following command to modify the MPI/util/Makefile file:

1. `vi MPI/util/Makefile`
   2. Press i to enter the editing mode and modify lines 3, 9, and 11 in the MPI/util/Makefile file. Pay attention to the information in bold.
      
      ```
      MPI_INST = /path/to/OPENMPI
      # LIB = -L$(MPI_INST)/lib -lmpi
      # CFLAGS = -mmodel=small -O2 -l $(INC)
      ```
      3. Press Esc, enter `.wq!`, and press Enter to save the file and exit.

**Step 6** Run the following commands to compile and install the software:

```
maker COMPILER=gfortranomp MPI=openmpi -j20
```

After the compilation is successful, the `CAMx_v6.50.openMPI.gfortranomp` executable file is generated in the src directory. You can run the `ls` command to view the file.

After the compilation is successful, the `CAMx_v6.50.openMPI.gfortranomp` executable file is generated in the `/path/to/CAMX/CAMX_v6-50` directory. You can run the `ls /path/to/CAMX/CAMX_v6-50` command to view the file. **Figure 1-1** shows the command output.

**Figure 1-1** Result example

```
```

-----End
1.2.7 Running and Verifying BWA

Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following commands to decompress the compute instance files:

```
cd /path/to/CASE

tar xzvf v6-50-specific-inputs-180430.tgz

tar xzvf CAMx6-x-test_run-inputs_met-140402.tgz

tar xzvf CAMx6-30-test_run-inputs_other-160408.tgz
```

**Step 3** Run the following commands to copy the test script:

```
cd runfiles

cp CAMx_v6.50.midwest.36.12.20020506-07.MPICH2.job openmpi4.job
```

**Step 4** Run the following command to modify the test script:

1. `vi openmpi4.job`

2. Press `I` to enter the insert mode and modify the test script. Pay attention to the information in bold.

   ```
   set EXEC      = "/path/to/CAMX/CAMX_v6-50/CAMx.v6.50.openMPI.gfortranomp"
   cat << ieof > nodes
   192.168.47.111
   set NUMPROCS = 96
   #mpdboot --n $RING --f nodes --verbose
   if( ![ mpirun --allow-run-as-root --machinefile nodes -np
       NUMPROCS SEXEC ] ) then
     exit
   endif
   exit
   endif
   ```

3. Press `Esc`, enter `.wq!`, and press `Enter` to save the file and exit.

**Step 5** Run the compute instances.

```
./openmpi4.job
```

Check the value of **days/ns** in the last **Info: Benchmark time** in the log. The unit is **days/ns**. A smaller value indicates better performance. You can also convert the value to the value of **ns/days**. A higher value indicates better performance.

**Figure 1-2** shows the command output.
1.2.8 More Information

Official Arm website:

1.3 CESM 2.1.1 Porting Guide (CentOS 7.6)

1.3.1 Introduction

The Community Earth System Model (CESM) is a fully coupled numerical simulation of the Earth system consisting of atmospheric, ocean, ice, land surface, carbon cycle, and other components. CESM includes a climate model providing state-of-art simulations of the Earth's past, present, and future. It is the successor of the Community Climate System Model (CCSM). CESM1 was released in 2010 with primary development by the Climate and Global Dynamics Division (CGD) of the National Center for Atmospheric Research (NCAR), and significant funding by the National Science Foundation (NSF) and the Department of Energy (DoE).

For more information about Bifrost, visit the official CESM website.

Programming language: Fortran

Brief description: a simulation of the Earth system

Open-source protocol: user-defined open-source protocol

Recommended Version

CESM 2.1.1

1.3.2 Environment Requirements

Hardware Requirements

Table 1-11 lists the hardware requirements.
Table 1-11 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 1-12 lists the software requirements.

Table 1-12 Software Requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CESM</td>
<td>2.1.1</td>
<td><a href="https://github.com/ESCOMP/cesm/releases/tag/release-cesm2.1.1">https://github.com/ESCOMP/cesm/releases/tag/release-cesm2.1.1</a></td>
</tr>
<tr>
<td>OpenBLAS</td>
<td>0.3.6</td>
<td><a href="https://github.com/xianyi/OpenBLAS/releases">https://github.com/xianyi/OpenBLAS/releases</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.1</td>
<td><a href="http://hdfgroup.org/HDF5/">http://hdfgroup.org/HDF5/</a></td>
</tr>
<tr>
<td>NETCDF</td>
<td>4.4.1.1</td>
<td><a href="https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1">https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1</a></td>
</tr>
<tr>
<td>NETCDF-F</td>
<td>4.4.1</td>
<td><a href="https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1">https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 1-13 lists the OS requirements.

Table 1-13 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

1.3.3 Planning the Paths for Software Porting

Table Paths for software porting lists the software installation paths involved in the CESM software porting.
Table 1-14 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/PNETCDF</td>
<td>Installation path of PnetCDF</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NetCDF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installation path of NETCDF-F</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/CESM</td>
<td>Installation path of CESM</td>
<td></td>
</tr>
</tbody>
</table>

1.3.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-15 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install OpenBLAS.</td>
<td>For details, see 1.3.4.1 Installing OpenBLAS.</td>
</tr>
<tr>
<td>3</td>
<td>Install HDF5.</td>
<td>For details, see 1.3.4.2 Installing HDF5.</td>
</tr>
<tr>
<td>No.</td>
<td>Operation</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>4</td>
<td>Install PnetCDF.</td>
<td>For details, see 1.3.4.3 Installing PnetCDF.</td>
</tr>
<tr>
<td>5</td>
<td>Install NetCDF.</td>
<td>For details, see 1.3.4.4 Installing NetCDF.</td>
</tr>
<tr>
<td>6</td>
<td>Install NetCDF-F.</td>
<td>For details, see 1.3.4.5 Installing NetCDF-F.</td>
</tr>
</tbody>
</table>

### 1.3.4.1 Installing OpenBLAS

**Procedure**

1. **Step 1** Use PuTTY to log in to the server as the root user.
2. **Step 2** Run the following command to decompress the OpenBLAS installation package:
   
   ```bash
   tar xvf OpenBLAS-0.3.6.tar.gz
   ```
3. **Step 3** Run the following command to switch to the directory generated after the package is decompressed:
   
   ```bash
   cd OpenBLAS-0.3.6
   ```
4. **Step 4** Run the following commands to perform configuration:
   
   ```bash
   export CC=`which gcc`
   export CXX=`which g++`
   export FC=`which gfortran`
   ```
5. **Step 5** Run the following commands to perform compilation and installation:
   
   ```bash
   make
   make PREFIX=/path/to/OPENBLAS install
   ----End
   ```

### 1.3.4.2 Installing HDF5

**Procedure**

1. **Step 1** Use PuTTY to log in to the server as the root user.
2. **Step 2** Run the following command to decompress the HDF5 installation package:
   
   ```bash
   tar -xvf hdf5-1.10.1.tar.gz
   ```
3. **Step 3** Run the following command to switch to the directory generated after the package is decompressed:
   
   ```bash
   cd hdf5-1.10.1
   ```
Step 4 Run the following commands to configure the environment before compilation:

```bash
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
```

Step 5 Run the following command to perform configuration:

```bash
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 6 Run the following commands to perform compilation and installation:

```bash
make -j 16
make install
```

---End

1.3.4.3 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the PnetCDF installation package:

```bash
tar -xvf parallel-netcdf-1.9.0.tar.bz2
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd parallel-netcdf-1.9.0
```

Step 4 Run the following commands to configure the environment before compilation:

```bash
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
```

Step 5 Run the following command to perform configuration:

```bash
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC" CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 6 Run the following commands to perform compilation and installation:

```bash
make -j 16
make install
```

----End
1.3.4.4 Installing NetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF installation package:

```
tar -xvf netcdf-4.4.1.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-4.4.1.1
```

Step 4 Run the following commands to configure the environment before compilation:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/PNETCDF/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -LWl,-rpath=/path/to/HDF5/lib -LWl,-rpath=/path/to/PNETCDF/lib"
export CFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -I/path/to/HDF5/include -I/path/to/PNETCDF/include"
```

Step 5 Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile CC=MPICC CXX=MPICXX FC=MPIFC F77=MPIFC
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

1.3.4.5 Installing NetCDF-F

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar -xvf netcdf-fortran-4.4.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-fortran-4.4.1
```
Step 4 Run the following commands to configure the environment before compilation:

```bash
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib"
export CFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
export CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
export FCFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
```

Step 5 Run the following command to perform configuration:

```bash
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 6 Run the following commands to perform compilation and installation:

```bash
make -j 16
make install
```

Step 7 Run the following command to clear compilation parameters:

```bash
unset CC CXX FC F77 CPPFLAGS LDFLAGS CFLAGS CXXFLAGS FCFLAGS
```

```
1.3.5 Obtaining the Source Code

Procedure

Step 1 Download the CESM installation package cesm-release-cesm2.1.1.tar.gz.

URL: https://github.com/ESCOMP/CESM/tags?after=cesm2.1-nemo.v01

Step 2 Use SFTP to upload the CESM installation package to the /path/to/CESM directory on the server.

```

1.3.6 Compiling and Installing CESM

Prerequisites

Configure an external network so that servers can access the Internet network.
Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the CESM installation package and go to the directory generated after the package is decompressed:

```
tar -xvf cesm-release-cesm2.1.1.tar.gz
cd cesm-release-cesm2.1.1
```

Step 3 Run the following command to install environment-modules, git, and svn:

```
yum install git svn environment-modules -y
```

Step 4 Run the following command to add verification information:

```
svn ls https://svn-ccsm-models.cgd.ucar.edu/ww3/release_tags
```

Step 5 Run the following command to install other CESM components:

```
./manage_externals/checkout_externals
```

Step 6 Run the following command to check the component installation: (If the following information is displayed, the installation is successful.)

```
./manage_externals/checkout_externals -S
```

```
Processing externals description file: Externals.cfg
Processing externals description file: Externals_CLM.cfg
Processing externals description file: Externals_POP.cfg
Processing externals description file: Externals_CISM.cfg
Checking status of externals: clm, fates, ptdm, mosart, ww3, cime, cice, pop, cvmix, marbl, cism, source_cism, rtm, cam,
./cime
./components/cam
./components/cice
./components/cism
./components/cism/source_cism
./components/clm
./components/clm/src/fates
./components/clm/tools/PTCLM
./components/mosart
./components/pop
./components/pop/externals/CVMix
./components/pop/externals/MARBL
./components/rtm
./components/rtm
```

Step 7 Run the following command to modify the config_machines.xml file:

1. `vi cime/config/cesm/machines/config_machines.xml`
2. Press I to enter the editing mode and modify lines 153, 170, 171, and 176 in the config_machines.xml file. Pay attention to the information in bold.

```
153 <NODENAME_REGEX>node1</NODENAME_REGEX>
170 <MAX_TASKS_PER_NODE>96</MAX_TASKS_PER_NODE>
171 <MAX_MPITASKS_PER_NODE>96</MAX_MPITASKS_PER_NODE>
176 <arg name="ntasks">--allow-run-as-root --mca btl ^openib -np \{ { total_tasks } } </arg>
```
3. Press Esc, enter `wq!`, and press Enter to save the file and exit.

----End
1.3.7 Running and Verifying CESM

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to switch to the working directory:

```
    cd /path/to/CESM/cesm-release-cesm2.1.1/cime/scripts
```

Step 3 Run the following command to create a case:

```
    ./create_newcase --case mycase --compset X --res f19_g16
```

Step 4 Run the following commands to download the import data:

```
    cd mycase
    ./check_case
    ./check_input_data --download
```

Step 5 Run the following commands to load environment variables:

```
    export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
    export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:/path/to/OPENBLAS/lib:/path/to/HDF5/lib:/path/to/PNETCDF/lib:/path/to/NETCDF/lib:$LD_LIBRARY_PATH
    export NETCDF_PATH=/path/to/NETCDF
```

Step 6 Run the following commands to start the CESM program:

```
    ./case.setup
    ./case.build
    ./case.submit
```

If the information shown in Figure 1-3 is displayed at the end of the OUTCAR log generated by CESM, the CESM program ends normally.

View the logs displayed on the screen, as shown in the red line in Figure 1-3. You need to manually calculate the time difference. A smaller value indicates better performance.
1.3.8 More Information

GitHub developer forum: [https://esmci.github.io/cime/index.html](https://esmci.github.io/cime/index.html)

1.4 NEMO V3.6 Porting Guide (CentOS 7.6)

1.4.1 Introduction

NEMO (Nucleus for European Modeling of the Ocean) is a state-of-the-art modelling framework for research activities and forecasting services in ocean and climate sciences.

For more information about NEMO, visit the [official NEMO website](https://www.nemo-ocean.eu/).

Programming language: C++

Brief description: Modelling framework for research activities and forecasting services in ocean and climate sciences.

Open-source protocol: Artistic License 2.0

Recommended Version

NEMO V3.6

1.4.2 Environment Requirements

Hardware Requirements

*Table 1-16* lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
Software Requirements

*Table 1-17* lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>XIOS</td>
<td>1.0</td>
<td><a href="http://forge.ipsl.jussieu.fr/ioserver/svn/XIOS/branches/xios-1.0">http://forge.ipsl.jussieu.fr/ioserver/svn/XIOS/branches/xios-1.0</a></td>
</tr>
<tr>
<td>Blitz</td>
<td>1.0.1</td>
<td><a href="https://github.com/blitzpp/blitz.git">https://github.com/blitzpp/blitz.git</a></td>
</tr>
<tr>
<td>PNETCDF</td>
<td>1.9.0</td>
<td><a href="http://cucis.ece.northwestern.edu/projects/PnetCDF">http://cucis.ece.northwestern.edu/projects/PnetCDF</a></td>
</tr>
<tr>
<td>Netcdf</td>
<td>4.4.1.1</td>
<td><a href="https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1">https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1</a></td>
</tr>
<tr>
<td>NETCDF-F</td>
<td>4.4.1</td>
<td><a href="https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1">https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.1</td>
<td><a href="https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/">https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/</a></td>
</tr>
</tbody>
</table>

OS Requirements

*Table 1-18* lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

1.4.3 Paths for Software Porting

This chapter lists the software installation paths involved in the NEMO software porting.
### Table 1-19 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Paths for Software Porting in Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/PNETCDF</td>
<td>Installation path of Pnetcdf</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NetCDF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installation path of NetCDF-Fortran</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to BLITZ</td>
<td>Installation path of Blitz</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>/path/to/XIOS</td>
<td>Installation path of XIOS</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>/path/to/NEMO</td>
<td>Installation path of NEMO</td>
<td></td>
</tr>
</tbody>
</table>

### 1.4.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.

**Procedure**

Table 1-20 Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing Pnetcdf</td>
<td>For details, see 1.4.4.1 Installing Pnetcdf.</td>
</tr>
</tbody>
</table>
1.4.4.1 Installing Pnetcdf

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install Pnetcdf.

```
tar -xvf parallel-netcdf-1.9.0.tar.bz2
cd parallel-netcdf-1.9.0
export MPICC=`which mpicc`
export MPICXX=`which mpicxx`
export MPIFC=`which mpifort`
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu
CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC"
CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
make -j 16
make install
```

Step 3 Run the following commands to load environment variables:

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.4.4.2 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following commands to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.5.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd hdf5-1.10.1
```

Step 4 Run the following command to perform configuration:

```
export MPICC=`which mpicc`
export MPICXX=`which mpicxx`
export MPIFC=`which mpifort`
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

Step 6 Run the following commands to load environment variables:

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

1.4.4.3 Installing NetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the NetCDF installation package:

```
tar -xvf netcdf-c-4.4.1.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-c-4.4.1.1
```

Step 4 Run the following command to perform configuration:

```
export MPICC=`which mpicc`
export MPICXX=`which mpicxx`
export MPIFC=`which mpifort`
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/PNETCDF/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -Wl,-rpath=/path/to/HDF5/install/lib"
```
export CFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -I/path/to/HDF5/include -I/path/to/PNETCDF/include"

./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile CC=SMPICC CXX=SMPIcxx FC=SMPIFC F77=SMPIFC

**Step 5** Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

**Step 6** Run the following commands to load environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

### 1.4.4.4 Installing NetCDF-Fortran

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the NetCDF-Fortran installation package:

```
tar -xvf netcdf-fortran-4.4.1.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-fortran-4.4.1
```

**Step 4** Run the following command to perform configuration:

```
export MPICC=`which mpicc`
export MPICXX=`which mpicxx`
export MPIFC=`which mpifort`
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib "
export CFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
export CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
export FCFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
```
Step 5 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

Step 6 Run the following commands to load environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```
Step 2  Run the following command to create the XIOS installation directory:

```
mkdir -p /path/to/XIOS
```

Step 3  Run the following command to switch to the directory generated after the package is decompressed:

```
cd /path/to/XIOS
```

Step 4  Run the following command to download the XIOS source code:

```
svn co -r 703 http://forge.ipsl.jussieu.fr/ioserver/svn/XIOS/branchs/xios-1.0
```

**NOTE**
Before running this command, ensure that the server can access the Internet.

Step 5  Run the following commands to install the system dependency libraries:

```
yum install -y perl-lib* boost-*
```

Step 6  Run the following command to switch to the directory generated after the package is decompressed:

```
cd xios-1.0
```

Step 7  Run the following commands to modify the `bld.cfg` file:

1. `vi bld.cfg`
2. Press **I** to enter the insert mode and modify the `bld.cfg` file. Pay attention to the information in bold.

   ```
   search_src true
   src::zzz
   #src::date SPWD/extern/boost/src/date_time
   src::blitz SPWD/extern/blitz/src
   src::netcdf SPWD/extern/netcdf4
   bld::lib xios
   bld::target libxios.a
   #bld::target generate_fortran_interface.exe
   #bld::target xios_server.exe test_client.exe parse_xml.exe test_complete.exe
   test_xios_interface.exe
   bld::exe_dep
   ```
3. Press **Esc**, enter `.wq!`, and press **Enter** to save the file and exit.

Step 8  Run the following command to create and edit the `arch-AARCH64_GNU_LINUX.env` configuration file:

1. `vi arch/arch-AARCH64_GNU_LINUX.env`
2. Press **I** to enter the editing mode and create and edit the `arch-AARCH64_GNU_LINUX.env` file.

```
# Edit the file based on the actual environment variables and dependency library installation path.
export HDF5_INC_DIR=’/path/to/HDF5/include’
export HDF5_LIB_DIR=’/path/to/HDF5/lib’
export NETCDF_INC_DIR=’/path/to/NETCDF/include’
export NETCDF_LIB_DIR=’/path/to/NETCDF/lib’
export BOOST_INC_DIR=’/usr/include/boost’
export BOOST_LIB_DIR=’/usr/lib64’
```
export BLITZ_INC_DIR="/path/to/BLITZ/include"
export BLITZ_LIB_DIR="/path/to/BLITZ/lib"

3. Press **Esc**, enter `.wq!`, and press **Enter** to save the file and exit.

**Step 9** Run the following command to create and edit the `arch-AARCH64_GNU_LINUX.fcm` configuration file:

1. `vi arch/arch-AARCH64_GNU_LINUX.fcm`
2. Press **I** to enter the editing mode and create and edit the `arch-AARCH64_GNU_LINUX.fcm` file.
   
   Edit the file based on the actual environment variables and dependency library installation path.
   
   ```
   #########################################################################
   #                      Projet xios - xmlserver                          #
   #########################################################################
   #########################################################################
   %CCOMPILER     mpicc
   %FCOMPILER     mpif90
   %LINKER        mpif90
   %BASE_CFLAGS  -ansi -w -D_GLIBCXX_USE_CXX11_ABI=0
   %PROD_CFLAGS   -O3 -DBOOST_DISABLE_ASSERTS
   %DEV_CFLAGS    -g -O2
   %DEBUG_CFLAGS  -g
   %BASE_FFLAGS   -D__NONE__
   %PROD_FFLAGS   -O3
   %DEV_FFLAGS    -g -O2
   %DEBUG_FFLAGS  -g
   %BASE_INC      -D__NONE__
   %BASE_LD      -lstdc++
   %CPP          cpp
   %FPP          cpp -P
   %MAKE         gmake
   
   3. Press **Esc**, enter `.wq!`, and press **Enter** to save the file and exit.

**Step 10** Run the following command to create and edit the `arch-AARCH64_GNU_LINUX.path` configuration file:

1. `vi arch/arch-AARCH64_GNU_LINUX.path`
2. Press **I** to enter the editing mode and create and edit the `arch-AARCH64_GNU_LINUX.path` file.
   
   Edit the file based on the actual environment variables and dependency library installation path.
   
   ```
   NETCDF_INCDIR="-I /path/to/NETCDF/include"
   NETCDF_LIBDIR="-L /path/to/NETCDF/lib"
   NETCDF_LIB="-lncdf -lnetcdff"
   
   MPI_INCDIR="-I /path/to/OPENMPI/include"
   MPI_LIBDIR="-L /path/to/OPENMPI/lib"
   MPI_LIB="-lmpi"
   
   HDF5_INCDIR="-I /path/to/HDF5/include"
   HDF5_LIBDIR="-L /path/to/HDF5/lib"
   HDF5_LIB="-lhdff5 -lhdff5_hl -lz"
   
   BOOST_INCDIR="-I /usr/include/boost/
   BOOST_LIBDIR="-L /usr/lib64/"
   BOOST_LIB=""```
3. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 11** Run the following commands to perform the first compilation and installation:

```bash
cd /path/to/XIOS/xios-1.0
chmod +x ./make_xios

cd extern
ln -s /path/to/BLITZ ./blitz
ln -s /path/to/NETCDF ./netcdf4
ln -s /usr/include/boost ./boost

cd ..
./make_xios --dev --job 32 --full --arch AARCH64_GNU_LINUX
```

**Step 12** Run the following commands to perform the second compilation and installation:

If an error is reported after the first compilation, run the following command again:

```bash
cd extern
rm -rf ./blitz
rm -rf ./netcdf4
rm -rf ./boost
ln -s /path/to/BLITZ ./blitz
ln -s /path/to/NETCDF ./netcdf4
ln -s /usr/include/boost ./boost

cd ..
./make_xios --dev --job 32 --full --arch AARCH64_GNU_LINUX
```

----End

### 1.4.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the NEMO installation package `NEMO_releases_release-3.6-10083.zip`.

Download it from [http://www.nemo-ocean.eu/](http://www.nemo-ocean.eu/).

**Step 2** Use SFTP to upload the NEMO installation package to the `/path/to/NEMO` directory on the server.

----End
1.4.6 Compiling and Installing NEMO

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to create and go to the NEMO installation directory:

```
mkdir -p /path/to/NEMO

cd /path/to/NEMO
```

Step 3 Run the following commands to configure the NEMO environment variables:

```
echo "export NETCDF_DIR=/path/to/NETCDF" >> nemo-env.sh

echo "export HDF_DIR=/path/to/HDF5" >> nemo-env.sh

echo "export XIOS_DIR=/path/to/XIOS/xios-1.0" >> nemo-env.sh

source nemo-env.sh
```

Step 4 Run the following command to decompress the installation package:

```
unzip NEMO_releases_release-3.6-10083.zip
```

Step 5 Run the following command to go to the NEMO source code directory:

```
cd NEMO/releases/release-3.6/NEMOGCM
```

Step 6 Run the following commands to create the arch-aarch64_gnu.fcm file:

1. vi ARCH/arch-aarch64_gnu.fcm
2. Press I to enter the editing mode and add the following content:

```bash
#generic gfortran compiler options for linux
#NCDF_INC   netcdf include file
#NCDF_LIB   netcdf library
#FC         Fortran compiler command
#FCFLAGS    Fortran compiler flags
#FFLAGS     Fortran 77 compiler flags
#LD         linker
#LDFLAGS    linker flags, e.g. -L<lib dir> if you have libraries in a
#FPFLAGS    pre-processing flags
#AR         assembler
#ARFLAGS    assembler flags
#MK         make
#USER_INC   additional include files for the compiler, e.g. -I<include dir>
#USER_LIB   additional libraries to pass to the linker, e.g. -L<library>
#CC         C compiler used to compile conv for AGRIF
#CFLAGS     compiler flags used with CC
#
#Note that:
# - unix variables "$..." are accepted and will be evaluated before calling fcm.
# - fcm variables are starting with a % (and not a $)
%NCDF_HOME   $NETCDF_DIR
%HDF5_HOME   $HDF_DIR
%XIOS_HOME   $XIOS_DIR

%NCDF_INC   -I%NCDF_HOME/include -I%HDF5_HOME/include
%NCDF_LIB   -L%HDF5_HOME/lib -L%NCDF_HOME/lib -lnetcdf -lnetcdff
%XIOS_INC   -I%XIOS_HOME/inc
%XIOS_LIB   -L%XIOS_HOME/lib -lxios
%CPP        cpp -Dkey_nosignedzero
%FC         mpif90 -c -cpp
```
%FCFLAGS  -mcpu=native -fdefault-real-8 -fdefault-double-8 -O3 -funroll-all-loops -fcray-pointer -ffree-line-length-none -g
%FFLAGS    %FCFLAGS
%LD        mpi90
%LDFLAGS    -Istdc++
%FPFFLAGS   -P -C -traditional
%AR        ar
%ARFLAGS    rs
%MK        gmake
%USER_INC   %XIOS_INC %NCDF_INC
%USER_LIB   %XIOS_LIB %NCDF_LIB

%CC        cc
%FFLAGS    -O0

NOTE

Ensure that no space exists at the end of a line or empty line.

3. Press Esc, type :wq!, and press Enter to save the file and exit.

Step 7 Run the following commands to compile and install the software:

chmod 777 ./ -R

cd ./CONFIG/

./makenemo -m aarch64_gnu -j 32 -r AMM12 -n 'MY_AMM12' add_key
"key_nosignedzero"

----End

1.4.7 Running and Verifying NEMO

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the work directory:

cd /path/to/NEMO/NEMO/releases/release-3.6/NEMOGCM/CONFIG/MY_AMM12/EXP00

Step 3 Upload AMM12_v3.6.tar to /path/to/NEMO/NEMO/releases/release-3.6/NEMOGCM/CONFIG/MY_AMM12/EXP00.

Step 4 Run the following commands to decompress the package:

tar xf AMM12_v3.6.tar

gunzip *.gz */*.gz

rm domain_def.xml field_def.xml namelist_ref -f

ln -s .././SHARED/domain_def.xml ./domain_def.xml

ln -s .././SHARED/field_def.xml ./field_def.xml

ln -s .././SHARED/namelist_ref ./namelist_ref

Step 5 Run the following command to create the host file:

1. vi hostfile
2. Press `I` to enter the editing mode and add the following content:

```
node1
node2
```

3. Press `Esc`, type `:wq!`, and press `Enter` to save the file and exit.

**Step 6** Run the following command to start the NEMO program.

```
{ time mpirun --allow-run-as-root -np 192 -N 96 -hostfile hostfile --mca btl
^openib ./opa ; } 2>&1 | tee -a NEMO.log
```

**Table 1-21 Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-np</td>
<td>Total number of processes used in the test.</td>
</tr>
<tr>
<td>-N</td>
<td>Number of processes running on each server.</td>
</tr>
<tr>
<td>--hostfile</td>
<td>Name of the node to be used.</td>
</tr>
</tbody>
</table>

If the information shown in **Figure 1-4** is displayed at the end of the `ocean.output` log generated by NEMO, the NEMO program ends normally.

**Figure 1-4** Test example

Check the value of `real` in the `NEMO.log` file. The unit is second. A higher the value indicates a lower performance. **Figure 1-5** shows the output result.
1.5 ROMS 3.6 Porting Guide (CentOS 7.6)

1.5.1 Introduction

The Regional Ocean Modeling System (ROMS) is a 3D regional ocean modeling system developed by the Rutgers University, University of California Los Angeles. It is widely used to simulate the hydrodynamic and water environment in the ocean and estuary.

For more information about the ROMS, visit the official ROMS website.

Programming language: Fortran

Brief description: 3D regional ocean modeling system

Open-source protocol: MIT/X license

Recommended Version

ROMS 3.6

1.5.2 Environment Requirements

Hardware Requirements

Table 1-22 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
Software Requirements

Table 1-23 lists the software requirements.

Table 1-23 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROMS</td>
<td>3.6</td>
<td><a href="https://www.myroms.org/svn/src/trunk">https://www.myroms.org/svn/src/trunk</a></td>
</tr>
<tr>
<td>PNETCFD</td>
<td>1.11.2</td>
<td><a href="http://cucis.ece.northwestern.edu/">http://cucis.ece.northwestern.edu/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>projects/PnetCDF/Release/pnetcdf-1.11.2.tar.gz</td>
</tr>
<tr>
<td>NETCDF-C</td>
<td>4.7.0</td>
<td><a href="https://github.com/Unidata/netcdf-c/">https://github.com/Unidata/netcdf-c/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>releases/tag/v4.7.0</td>
</tr>
<tr>
<td>NETCDF-Fortran</td>
<td>4.4.5</td>
<td><a href="https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.5">https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.5</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.5</td>
<td><a href="https://portal.hdfgroup.org/display/">https://portal.hdfgroup.org/display/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>support/HDF5+1.10.5#files</td>
</tr>
<tr>
<td>Test case</td>
<td>ocean_upwelling</td>
<td>Calculation case provided by the software</td>
</tr>
</tbody>
</table>

OS Requirements

Table 1-24 lists the OS requirements.

Table 1-24 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

1.5.3 Planning the Paths for Software Porting

Table Paths for software porting lists the software installation paths involved in the ROMS software porting.
### Table 1-25 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/PNETCDF</td>
<td>Installation path of PnetCDF</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NetCDF-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installation path of NetCDF-Fortran</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/ROMS</td>
<td>Installation path of ROMS</td>
<td></td>
</tr>
</tbody>
</table>

### 1.5.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see section &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Create an environment variable file.</td>
<td>For details, see 1.5.4.1 Creating an Environment Variable File.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Install HDF5.</td>
<td>For details, see 1.5.4.2 Installing HDF5.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Install PnetCDF.</td>
<td>For details, see 1.5.4.3 Installing PnetCDF.</td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Operation</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Install NetCDF-C.</td>
<td>For details, see 1.5.4.4 Installing NetCDF-C.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Install NetCDF-Fortran.</td>
<td>For details, see 1.5.4.5 Installing NetCDF-Fortran.</td>
<td></td>
</tr>
</tbody>
</table>

### 1.5.4.1 Creating an Environment Variable File

#### Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to create the env.sh environment variable file:

1. vi /root/env.sh
2. Press i to enter the editing mode and edit the env.sh file.

```bash
#!/bin/bash
export GNU_HOME=
export MPI_HOME=
export HDF5=
export PNETCDF=
export NETCDF=
export PHDF5=$HDF5
export CPPFLAGS=" -I$HDF5/include -I$PNETCDF/include -I$NETCDF/include -I$MPI_HOME/
include 
export CFLAGS=" -I$HDF5/include -I$PNETCDF/include -I$NETCDF/include -I$MPI_HOME/include 
export CXXFLAGS=" -I$HDF5/include -I$PNETCDF/include -I$NETCDF/include -I$MPI_HOME/include 
export FCFLAGS=" -I$HDF5/include -I$PNETCDF/include -I$NETCDF/include -I$MPI_HOME/include 
export FFLAGS=" -I$HDF5/include -I$PNETCDF/include -I$NETCDF/include -I$MPI_HOME/include 
export PATH=$NETCDF/bin:$PNETCDF/bin:$MPI_HOME/bin:$GNU_HOME/bin:$PATH
```

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 3** Run the following command to make the configuration file take effect:

```bash
source /root/env.sh
```

----End

### 1.5.4.2 Installing HDF5

#### Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the HDF5 installation package:

```
tar zxfv hdf5-1.10.1.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:
cd hdf5-1.10.1

Step 4 Run the following command to perform configuration:

```
./configure --prefix=$HDF5 --build=aarch64-linux --enable-parallel --enable-shared --enable-fortran CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 5 Run the following commands to compile and install HDF5:

```
make -j 16
make install
```

1.5.4.3 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the PnetCDF installation package:

```
tar zxvf pnetcdf-1.11.2.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd pnetcdf-1.11.2
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=$PNETCDF --build=aarch64-linux --enable-shared --enable-fortran --enable-large-file-test CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 5 Run the following commands to compile and install PnetCDF:

```
make -j 16
make install
```

1.5.4.4 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the NetCDF-C installation package:

```
tar zxvf netcdf-c-4.7.0.tar.gz
yum install curl.aarch64 libcurl.aarch64 libcurl-devel.aarch64 -y
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd netcdf-c-4.7.0
```
Step 4 Run the following command to perform configuration:

```
./configure --prefix=$NETCDF --build=aarch64-linux --enable-fortran --disable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-large-file-tests --enable-largefile CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 5 Run the following commands to compile and install NetCDF-C:

```
make -j 16
make install
```

1.5.4.5 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-Fortran installation package:

```
tar zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd netcdf-fortran-4.4.5
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=$NETCDF --build=aarch64-linux --disable-static --enable-shared --enable-parallel-tests --enable-large-file-tests --enable-largefile CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 5 Run the following commands to compile and install NetCDF-Fortran:

```
make -j 16
make install
```

1.5.5 Obtaining the Source Code

Procedure

Step 1 Download the ROMS installation package ROMS_3.6.

URL: https://www.myroms.org/

NOTE

To download the ROMS installation package, you need to register an account on the ROMS website and log in to the website using the registered account.
1.5.6 Compiling and Installing ROMS

**Procedure**

**Step 1** Use SVN to check out and obtain the ROMS source code.

Create a folder and check out the ROMS source code to the folder. To obtain the source code, you need to register an account.

```bash
cd /path/to/ROMS
mkdir ROMS_3.6
svn checkout --username Account --password Password https://www.myroms.org/svn/src/trunk ROMS_3.6
```

Example:

Run the following command to check out the source code using **HPC/Huawei12**:

```bash
svn checkout --username HPC --password Huawei12#$ https://www.myroms.org/svn/src/trunk ROMS_3.6
```

The source code file structure is as follows:
Step 2 Customize the build script.

1. To keep the source code unchanged, run the following command to create a ROMSProjects folder for compiling and running the ROMS. `upwelling` is the default example in the source code package.

   ```
   cd /path/to/ROMS
   mkdir ROMSProjects
   cd ROMSProjects
   mkdir upwelling
   ```

2. Run the following command to copy the `build.bash` file to the `upwelling` directory in the current environment. The `build.sh` and `build.bash` files exist in the ROMS/Bin directory of the source code.

   ```
   cp /path/to/ROMS/ROMS_3.6/ROMS/Bin/build.bash .
   ```

3. Run the following command to edit the `build.bash` file:

   a. `vi build.bash`
b. Press \texttt{i} to enter the insert mode and modify the file as follows:

\begin{verbatim}
export ROMS_APPLICATION=UPWELLING
export MY_PROJECT_DIR=/path/to/ROMSProjects/upwelling
export MY_ROMS_SRC=/path/to/ROMS/ROMS_3.6
export USE_MPI=on
export USE_MPIF90=on
export which_MPI=openmpi
export FORTран=gfortran
export USE_LARGE=on
export USE_NETCDF4=on
export USE_PARALLEL_IO=on
export PATH=/path/to/OPENMPI/bin:$PATH
export NF_CONFIG=/path/to/NETCDF/bin/nf-config
export NETCDF_INCDIR=/path/to/NETCDF/include
export NETCDF_LIBDIR=/path/to/NETCDF/lib
\end{verbatim}

**Step 3** Run the following commands to copy the input and configuration files:

\begin{verbatim}
cp /path/to/ROMS/ROMS_3.6/ROMS/External/ocean_upwelling.in .
cp /path/to/ROMS/ROMS_3.6/ROMS/External/varinfo.dat
cp /path/to/ROMS/ROMS_3.6/ROMS/Include/upwelling.h .
\end{verbatim}

**Step 4** Run the following command to edit the \texttt{ocean_upwelling.in} file:

1. \texttt{vi ocean_upwelling.in}
2. Press \texttt{i} to enter the insert mode and modify the \texttt{ocean_upwelling.in} file.

   Change the path whose \texttt{VARNAME} is \texttt{varinfo.dat}. Pay attention to the information in \textbf{bold}.

   \begin{verbatim}
   VARNAME = /path/to/ROMS/ROMSProjects/upwelling/varinfo.dat
   \end{verbatim}

   Set \texttt{NtileI} and \texttt{NtileJ}. The product of \texttt{NtileI} and \texttt{NtileJ} must be the same as the value of \texttt{np}. For example, if \texttt{np} is set to 96, you can set \texttt{NtileI} and \texttt{NtileJ} as follows (pay attention to the information in \textbf{bold}):

   \begin{verbatim}
   NtileI == 6
   NtileJ == 16
   \end{verbatim}

   \textbf{NOTE}

   For details about how to set \texttt{NtileI} and \texttt{NtileJ}, see 1.5.8 Troubleshooting.

3. Press \texttt{Esc}, enter \texttt{:wq!}, and press \texttt{Enter} to save the file and exit.

**Step 5** Compile ROMS.

\begin{verbatim}
./build.bash -j 8
\end{verbatim}

When MPI is used, the executable file \texttt{OceanM} is generated after successful compilation. You can run the \texttt{ls} command to view the file.

\begin{verbatim}
ll /path/to/ROMS/ROMSProjects/upwelling
\end{verbatim}

---End

**1.5.7 Runing and Verifying ROMS**

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.
Step 2  Run the following command to switch to the directory where the test case is stored:

```
cd /path/to/ROMS/ROMSProjects/upwelling
```

Step 3  Run the following command to start ROMS:

```
mpirun --allow-run-as-root --mca btl ^openib -np 96 ./oceanM ocean_upwelling.in > log 2>&1
```

The following files are generated: `ocean_avg.nc`, `ocean_dia.nc`, `ocean_his.nc`, and `ocean_rst.nc`.

Check the value of **Node# CPU: 12.334** in the log file. The unit is second. A smaller value indicates better performance.

**Figure 1-6** shows the command output.

**Figure 1-6** Result example

![Tile partition information for Grid 01: 0041x0080x0016 tiling: 006x016](image)

**Figure 1-7** shows an example of the CPU running time.

**Figure 1-7** Result example

![Tile partition information for Grid 01: 0041x0080x0016 tiling: 006x016](image)

---End

1.5.8 Troubleshooting

**Problem 1:** The format of the source code checked out in Windows is inconsistent with that in Linux.

**Symptom**
Kunpeng BoostKit for HPC
Porting Guide

1 Meteorology

The format of the source code checked out in Windows is different from that in
Linux.
Procedure
Step 1 Install and use dos2unix to convert the entire directory checked out by SVN in
Windows to the Linux format.
yum -y install dos2unix*
cd /home/hesj/tool/ROMS_916
find . -type f -exec dos2unix {} \;
----End

1.5.9 More Information
Wikipedia:
https://www.myroms.org/wiki/Getting_Started

1.6 WAVEWATCH III 6.07.1 Porting Guide (CentOS 7.6)
1.6.1 Introduction
WAVEWATCH III is a community wave modeling framework that includes the
latest scientific advances in wave modeling and dynamics.
WAVEWATCH III resolves the random-phase spectrum action balance equation of
the wavenumber direction spectrum. The model includes options for shallow
water (surfing area) applications and wetting and drying of grid points.
For details about WAVEWATCH III, visit the official WAVEWATCH III website.
Language: Fortran
Brief description: Third-generation wave mode.
Open-source protocol: user-defined open-source protocol.

Recommended Tool Version
WAVEWATCH III 6.07.1

1.6.2 Environment Requirements
Hardware Requirements
Table 1-27 lists the hardware requirements.

Issue 14 (2021-07-23)

Copyright © Huawei Technologies Co., Ltd.

51


Table 1-27 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 1-28 lists the software requirements.

Table 1-28 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDF5</td>
<td>1.10.1</td>
<td><a href="http://hdfgroup.org/HDF5/">http://hdfgroup.org/HDF5/</a></td>
</tr>
<tr>
<td>NetCDF</td>
<td>4.4.1.1</td>
<td><a href="https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1">https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1</a></td>
</tr>
<tr>
<td>NetCDF-FORTRAN</td>
<td>4.4.1</td>
<td><a href="https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1">https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1</a></td>
</tr>
<tr>
<td>PARMETIS</td>
<td>4.0.3</td>
<td><a href="http://glaros.dtc.umn.edu/gkhome/metis/parmetis/download">http://glaros.dtc.umn.edu/gkhome/metis/parmetis/download</a></td>
</tr>
<tr>
<td>Test case</td>
<td>mww3_data_01</td>
<td>ftp://polar.ncep.noaa.gov/tempor/ww3ftp/ww3_from_ftp.v6.07.tar.gz</td>
</tr>
</tbody>
</table>

OS Requirements

Table 1-29 lists the OS requirements.

Table 1-29 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

1.6.3 Planning the Paths for Software Porting

Table 1-30 lists the software installation paths involved in the WAVEWATCH III software porting.
Table 1-30 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each installation package for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NetCDF&lt;br&gt;Installation path of NetCDF-Fortran</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/METIS</td>
<td>Installation path of ParMETIS</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/WAVEWATCH</td>
<td>Installation path of WAVEWATCH III</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/CASE</td>
<td>Test path of WaveWATCH III</td>
<td></td>
</tr>
</tbody>
</table>

1.6.4 Configuring the Compilation Environment

Prerequisites

Use SFTP to upload the installation packages to the planned directories on the server.

Configuration Process

Table 1-31 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing HDF5</td>
<td>For details, see 1.6.4.1 Installing HDF5.</td>
</tr>
<tr>
<td>3</td>
<td>Installing NetCDF</td>
<td>For details, see 1.6.4.2 Installing NetCDF.</td>
</tr>
<tr>
<td>No.</td>
<td>Configuration Item</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>--------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>4</td>
<td>Installing NetCDF-Fortran</td>
<td>For details, see 1.6.4.3 Installing NetCDF-Fortran.</td>
</tr>
<tr>
<td>5</td>
<td>Installing ParMETIS</td>
<td>For details, see 1.6.4.4 Installing ParMETIS.</td>
</tr>
</tbody>
</table>

1.6.4.1 Installing HDF5

Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following commands to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.1.tar.gz
```

**Step 3** Run the following command to switch to the directory containing the decompressed files:

```
cd hdf5-1.10.1
```

**Step 4** Run the following command perform configuration:

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared
```

**Step 5** Run the following commands to perform compilation and installation:

```
make -j 20
make install
```

**Step 6** Run the following commands to load environment variables:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
export CC=$MPICC
export CXX=$MPICXX
export FC=$MPIFC
export F77=$MPIFC
export LD_LIBRARY_PATH=/path/to/HDF5/lib:SLD_LIBRARY_PATH
```

----End
1.6.4.2 Installing NetCDF

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the NetCDF installation package:

```bash
tar -xvf netcdf-4.4.1.1.tar.gz
```

**Step 3** Run the following command to switch to the directory containing the decompressed files:

```bash
cd netcdf-4.4.1.1
```

**Step 4** Run the following command perform configuration:

```bash
dir
```

**Step 5** Run the following commands to perform compilation and installation:

```bash
make -j 20
make install
```

---End

1.6.4.3 Installing NetCDF-Fortran

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the NetCDF-Fortran installation package:

```bash
tar -xvf netcdf-fortran-4.4.1.tar.gz
```

**Step 3** Run the following command to switch to the directory containing the decompressed files:

```bash
cd netcdf-fortran-4.4.1
```

**Step 4** Run the following command perform configuration:

```bash
dir
```

**Step 5** Run the following commands to perform compilation and installation:

```bash
make -j 20
make install
```

---End
Step 5 Run the following commands to perform compilation and installation:

```bash
make -j 20
make install
```

Step 6 Run the following command to delete the environment variables:

```bash
unset CC CXX FC F77 MPICC MPICXX MPIFC
```

---End

### 1.6.4.4 Installing ParMETIS

**Procedure**

Step 1 Use PuTTY to log in to the server as the **root** user.

Step 2 Run the following commands to decompress the ParMETIS installation package:

```bash
tar -xvf parmetis-4.0.3.tar.gz
```

Step 3 Run the following command to switch to the directory containing the decompressed files:

```bash
cd parmetis-4.0.3
```

Step 4 Run the following command to modify the `metis/include/metis.h` file:

1. `vi metis/include/metis.h`
2. Press **I** to enter the editing mode and modify line 33 in the `metis/include/metis.h` file
   ```bash
   #define IDXTYPEWIDTH 64
   ```
3. Press **Esc**, enter `:wq!`, and press **Enter** to save the script and exit.

Step 5 Run the following command to install CMake.

```bash
yum install cmake -y
```

Step 6 Run the following commands to go to the `metis` directory and compile and install METIS:

```bash
cd metis/
make config shared=1 prefix=/path/to/METIS
make install
```

Step 7 Run the following commands to go to the upper-layer directory and perform compilation and installation:

```bash
cd ../
make config shared=1 prefix=/path/to/METIS
make install
```

---End
1.6.5 Obtaining the Source Code

Procedure

Step 1 Download the WAVEWATCH III installation package WW3-6.07.1.tar.gz.
Download address: https://polar.ncep.noaa.gov/waves/wavewatch/.

Step 2 Use SFTP to upload the WAVEWATCH III installation package to the /path/to/WAVEWATCH directory on the server.

1.6.6 Compiling and Installing the Software

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to switch to the directory where WAVEWATCH III is installed:

cd /path/to/WAVEWATCH

Step 3 Run the following command to decompress the WAVEWATCH III source code package:

tar -xvf WW3-6.07.1.tar.gz

Step 4 Run the following command to switch to the directory containing the decompressed files:

cd WW3-6.07.1

Step 5 Run the following command to create the environment variable file:

1. vi env_ww3.sh
2. Press I to enter the editing mode and create the env_ww3.sh file.

```
# gnu + openmpi
export PATH=/path/to/GNU/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:$LD_LIBRARY_PATH

# netcdf
export NETCDF_DIR=/path/to/NETCDF
export NETCDF_LIBDIR=$NETCDF_DIR/lib
export NETCDF_INCDIR=$NETCDF_DIR/include
export PATH=$NETCDF_DIR/bin:$PATH
export LD_LIBRARY_PATH=$NETCDF_LIBDIR:$LD_LIBRARY_PATH

# metis
export METIS_PATH=/path/to/METIS
export METIS_PATH=$METIS_PATH/bin:$PATH
```
3. Press **Esc**, enter `:wq!`, and press **Enter** to save the script and exit.

### Step 6
Run the following command to load environment variables:

```
source env_ww3.sh
```

### Step 7
Run the following command to configure the WAVEWATCH III parameters (pay attention to the information in bold):

```
./model/bin/w3_setup model/
```

***************
*** WAVEWATCH III setup ***
***************

([INFO]) local env file wwatch3.env created in /home/WW3-6.07.1/model/bin/wwatch3.env
Setup file /home/WW3-6.07.1/model/bin/wwatch3.env not found

Default set up:
- Printer (listings) : **printer**
- Auxiliary FORTRAN comp. : **gfortran**
- Auxiliary C compiler : **gcc**
- Scratch directory : **/path/to/WAVEWATCH/WW3-6.07.1/model/tmp**
- Save sources : **yes**
- Save listings : **yes**
- Update settings ? [y/n] **y**

Creating new set-up:
- Printer for listings [**printer**] :
- Auxiliary FORTRAN compiler [**gfortran**] :
- Auxiliary C compiler [**gcc**] :
- Scratch space [**/home/wdy/WW3-6.07.1/model/tmp**] :
- Save source code files (*.f) [**yes**] :
- Save listing files [**yes**] :

Modified set up:
- Printer (listings) : **printer**
- Auxiliary FORTRAN comp. : **gfortran**
- Auxiliary C compiler : **gcc**
- Scratch directory : **/path/to/WAVEWATCH III/WW3-6.07.1/model/tmp**
- Save sources : **yes**
- Save listings : **yes**
- New settings OK ? [y/n] **y**

Setup makefile for auxiliary programs

Create required model subdirectories

Finished setting up WAVEWATCH III

----End

### 1.6.7 Running and Verifying the Software

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.
Step 2  Run the following commands to create and go to the working directory:

```bash
mkdir /path/to/WW3_TEST

cd /path/to/WW3_TEST
```

Step 3  Download the test case and copy it to the /path/to/WW3_TEST directory.

Step 4  Run the following commands to decompress the package:

```bash
tar -xvf ww3_from_ftp.v6.07.tar.gz

cd cases
```

Step 5  Run the following commands to set environment variables:

```bash
source /path/to/WAVEWATCH/WW3-6.07.1/env_ww3.sh
export ww3_dir=/path/to/WAVEWATCH/WW3-6.07.1/model
```

Step 6  Run the following commands to modify the mww3_case_01 file:

1. ```bash
vi mww3_case_01
```
2. Press I to enter the editing mode and modify the mww3_case_01 file.
   Line 61:
   ```bash
   #cd /scratch1/portfolios/NCEPDEV/ocean/noscrub/Henrique.Alves/WW3_CASES/cases
   ```
   Line 65:
   ```bash
   #module load intel mpt grads
   ```
   Line 72:
   ```bash
   compstr="Gnu"
   ```
   Line 143:
   ```bash
   #ww3_dir='echo $case_dir | sed 's//cases/*//g''
   ```
   Line 138:
   ```bash
   proc=96
   ```
   Line 420:
   ```bash
   mpirun --allow-run-as-root --mca btl ^openib -np $proc --hostfile /path/to/WW3_TEST/hostfile $path_e/ww3_multi
   ```
3. Press Esc, enter :wq!, and press Enter to save the script and exit.

Step 7  Run the following command to create the host file:

1. ```bash
vi hostfile
```
2. Press I to enter the editing mode and add the following content:
   ```bash
   node1
   node2
   ```
3. Press Esc, enter :wq!, and press Enter to save the script and exit.

Step 8  Open another login window and use PuTTY to log in to node 2 as the root user.

Step 9  On node 2, run the following command to configure environment variables:

1. ```bash
vi /root/.bashrc
```
2. Press i to enter the editing mode, and add the following content to the end of the .bashrc file:
   ```bash
   export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
   ```
3. Press Esc, enter :wq!, and press Enter to save the script and exit.
Step 10  On node 1, run the following command to start the WAVEWATCH III program:

`./mww3_case_01`

Step 11  If the information shown in Figure 1-8 is displayed when the WAVEWATCH III program is complete, the WAVEWATCH III program is running properly.

![Figure 1-8 Result example](image)

Check the value of Elapsed time in the `work_case_01/log.mww3` log file. The unit is second. A smaller value indicates better performance.

Figure 1-9 shows the command output.

![Figure 1-9 Result example](image)

1.6.8 Support and Other Resources

WAVEWATCH III quick installation guide:


1.7 Bifrost V1.1.1 Porting Guide (CentOS 7.6)

1.7.1 Introduction

Bifrost is a numerical model software for star convection and light bulbs.

For more information about Bifrost, visit the official Bifrost website.

Programming language: Fortran

Brief description: numerical model software for star convection and light bulbs

Open-source protocol: user-defined open-source protocol

Recommended Version

Bifrost V1.1.1
1.7.2 Environment Requirements

Hardware Requirements

Table 1-32 lists the hardware requirements.

Table 1-32 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 1-33 lists the software requirements.

Table 1-33 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bifrost</td>
<td>V1.1.1</td>
<td><a href="https://www.sigma2.no/procurement-project-hpc-b1">https://www.sigma2.no/procurement-project-hpc-b1</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 1-34 lists the OS requirements.

Table 1-34 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
</tbody>
</table>

1.7.3 Planning the Path for Software Porting

Table 1-35 lists the software installation paths involved in the Bifrost software porting.
Table 1-35 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/BIFROST</td>
<td>Installation path of Bifrost</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

1.7.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-36 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

1.7.5 Obtaining the Source Code

Procedure

Step 1  Download the Bifrost installation package Bifrost_bench_v1.1.1-sigma2.tgz.

URL: https://www.sigma2.no/procurement-project-hpc-b1

Step 2  Use SFTP to upload the Bifrost installation package to the /path/to/BIFROST directory on the server.

----End
1.7.6 Compiling and Installing Bifrost

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to decompress the Bifrost installation package:

```
cd /path/to/BIFROST

# Decompress the package

tar -xvf Bifrost_bench_v1.1.1-sigma2.tgz

cd Bifrost_bench_v1.1.1
```

Step 3  Run the following command to modify the Makefile file:

1. `vi Makefile`

2. Press `I` to enter the insert mode and modify the Makefile file. Pay attention to the information in bold.

```
# set operating system dependent parameters
F90     ?= mpifort
F77     = $(F90)
CC      = mpicc
LD      = $(F90)
F90FLAGS     ?= -O3 -ffree-line-length-0 -fno-range-check
F77FLAGS   = $(F90FLAGS)
LIBS     =
INCL     =
DEBUG    =
LDFLAGS  ?= $(F90FLAGS) $(DEBUG) $(INCL)
OUTP     += compiler
```

3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

Step 4  Run the following command to modify the mpi.f90 file:

1. `vi mpi.f90`

2. Press `I` to enter the insert mode and modify lines 776 and 778 in the mpi.f90 file. Pay attention to the information in bold.

```
# if(gridsize < 1000) then
fmti="(' ',I3)"
else
fmti="(' ',I4)"
endif

do k=dims(3)-1,0,-1
do j=dims(2)-1,0,-1
  fmt="(A1)"
do kk=0,(j-1)*2+5
    write (*,fmt,ADVANCE='NO') "  
  end do
  fmt=fmti
do i=0,dims(1)-1
  do n=0,gridsize-1
    if ((i.eq.(griddat(1,n)).and.(j.eq.griddat(2,n)).and.(k.eq.griddat(3,n))))
      write (*,fmt,ADVANCE='NO') n
    end do
  end do
```

3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

Step 5  Run the following command to perform compilation and installation:

```
make
```
**Step 6** Run the following command to view whether the `photo_tr.x` file is generated:

```
ll RUNS/photo_tr.x
```

```
-rwxr-xr-x 1 root root 2521960 Jan 11 14:39 RUNS/photo_tr.x
```

----End

## 1.7.7 Running and Verifying Bifrost

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to go to the `RUNS` directory:

```
cd /path/to/BIFROST/Bifrost_bench_v1.1.1/RUNS
```

**Step 3** Run the following command to create the `hostfile` file:

1. `vi hostfile`
2. Press `I` to enter the editing mode and add the following content:

   ```
   node1
   node2
   ```

   **NOTE**

   In the preceding command, `node1` and `node2` are the host names of the test servers. Change them based on the site requirements.

3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

**Step 4** Open another login window and use PuTTY to log in to node 2 as the **root** user.

**Step 5** On node 2, run the following command to configure environment variables:

1. `vi /root/.bashrc`
2. Press `i` to enter the editing mode, and add the following content to the end of the `.bashrc` file:

   ```
   export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
   ```

3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

**Step 6** On node 1, run the following command to start the Bifrost program:

```
mpirun --allow-run-as-root -np 192 -N 96 -hostfile hostfile photo_tr.x 2>&1 | tee -a Bifrost.out
```

**Table 1-37** Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-np</td>
<td>Total number of processes in use</td>
</tr>
<tr>
<td>-N</td>
<td>Number of processes running on each server</td>
</tr>
<tr>
<td>-hostfile</td>
<td>Host name of the node in use</td>
</tr>
</tbody>
</table>
Step 7 If the information shown in Figure 1-10 is displayed at the end of the generated log file `Bifrost.out` after 500 iterations, the Bifrost program is running properly.

Figure 1-10 Result example

Check the value of Mz/s (unit: Mz/s) in the `Bifrost.out` log. A greater value indicates better performance.

Figure 1-11 shows the command output.

Figure 1-11 Result example

---End

1.8 FVCOM 4.1 Porting Guide (CentOS 7.6)

1.8.1 Introduction

FVCOM is an unstructured-grid, finite-volume, free-surface, 3-D primitive equation coastal ocean circulation model. The model consists of momentum, continuity, temperature, salinity, and density equations and is closed physically and mathematically using turbulence closure submodels. The horizontal grid consists of unstructured triangular cells and the irregular bottom is presented using generalized terrain-following coordinates. The General Ocean Turbulent Model (GOTM) developed by Burchard's research group in Germany (Burchard, 2002) has been added to FVCOM to provide optional vertical turbulent closure schemes. FVCOM is solved numerically by a second-order accurate discrete flux calculation in the integral form of the governing equations over an unstructured triangular grid. This approach combines the best features of finite-element methods (grid flexibility) and finite-difference methods (numerical efficiency and code simplicity), and provides a much better numerical representation of both local and global momentum, mass, salt, heat, and tracer conservation. The ability of FVCOM to accurately solve scalar conservation equations in addition to the topological flexibility provided by unstructured meshes and the simplicity of the coding structure make FVCOM ideally suited for many coastal and interdisciplinary scientific applications.

For more information about FVCOM, visit the official FVCOM website.

Programming language: C/Fortran 90
Brief description: FVCOM is a finite-volume, free-surface, 3-D primitive equation coastal ocean circulation model comprised of unstructured triangular cells.

Open-source license: GPL 3.0

Recommended Version

FVCOM 4.1

1.8.2 Environment Requirements

Hardware Requirements

Table 1-38 lists the hardware requirements.

Table 1-38 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software requirements

Table 1-39 lists the software requirements.

Table 1-39 Software Requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>FVCOM</td>
<td>4.1</td>
<td><a href="http://fvcom.smast.umassd.edu/releases/fvcom-4.1.tar.gz">http://fvcom.smast.umassd.edu/releases/fvcom-4.1.tar.gz</a></td>
</tr>
<tr>
<td>metis</td>
<td>4.0</td>
<td>Provided by the software.</td>
</tr>
<tr>
<td>NetCDF-C</td>
<td>4.7.0</td>
<td><a href="https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.7.0">https://codeload.github.com/Unidata/netcdf-c/tar.gz/refs/tags/v4.7.0</a></td>
</tr>
<tr>
<td>NetCDF-Fortran</td>
<td>4.4.5</td>
<td><a href="https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.4.5">https://codeload.github.com/Unidata/netcdf-fortran/tar.gz/refs/tags/v4.4.5</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.5</td>
<td><a href="https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10.5.tar.gz">https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10.5.tar.gz</a></td>
</tr>
<tr>
<td>Test case</td>
<td>tst_run.nml</td>
<td>Provided by the software.</td>
</tr>
</tbody>
</table>

OS Requirements

Table 1-40 lists the OS requirements.
### 1.8.3 Planning the Paths for Software Porting

*Table 1-41* lists the software installation paths involved in the FVCOM software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package for setting up the basic environment</td>
<td>For details, see <em>Planning the Installation Paths</em> in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>4</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5</td>
<td>The planned installation paths are only examples. You are advised to deploy the software in shared paths. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.</td>
</tr>
<tr>
<td>5</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NetCDF-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installation path of NetCDF-Fortran</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/METIS</td>
<td>Path for storing the downloaded METIS patch</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>/path/to/FVCOM</td>
<td>Installation path of FVCOM</td>
<td></td>
</tr>
</tbody>
</table>

### 1.8.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.
Procedure

Table 1-42 Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section “Setting Up the Environment for the Cluster Scenario” in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Installing HDF5</td>
<td>See 1.8.4.1 Installing HDF5.</td>
</tr>
<tr>
<td>3</td>
<td>Installing NetCDF-C</td>
<td>See 1.8.4.2 Installing NetCDF-C.</td>
</tr>
<tr>
<td>4</td>
<td>Installing NetCDF-Fortran</td>
<td>See 1.8.4.3 Installing NetCDF-Fortran.</td>
</tr>
</tbody>
</table>

1.8.4.1 Installing HDF5

Procedure

**Step 1** Use PuTTY to log in to the server as the *root* user.

**Step 2** Run the following commands to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.5.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:

```
cd hdf5-1.10.5
```

**Step 4** Run the following command to perform the configuration:

```
./configure --prefix=/path/to/HDF5 --enable-fortran --enable-parallel CC=mpicc FC=mpif90 CXX=mpicxx
```

**Step 5** Run the following commands to perform compilation and installation:

```
make -j
make install
```

**Step 6** Run the following commands to set the HDF5 environment variables:

```
export LD_LIBRARY_PATH=/pat/to/HDF5/lib:SLD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End
1.8.4.2 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following command to decompress the NetCDF-C installation package:
```console
tar -zxvf netcdf-c-4.7.0.tar.gz
```
Step 3 Run the following command to switch to the directory generated after decompression:
```console
cd netcdf-c-4.7.0
```
Step 4 Run the following command to perform the configuration:
```console
./configure --prefix=/path/to/NETCDF LDFLAGS="-LShdf5/lib" CPPFLAGS="-I$hdf5/include" CC=mpicc --disable-dap
```
Step 5 Run the following commands to perform compilation and installation:
```console
make -j
make install
```
Step 6 Run the following commands to set the NetCDF-C environment variables:
```console
export PATH=/path/to/NETCDF/bin:SPATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:SLD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End

1.8.4.3 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following commands to decompress the NetCDF-Fortran installation package:
```console
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```
Step 3 Run the following command to switch to the directory generated after decompression:
```console
cd netcdf-fortran-4.4.5
```
Step 4 Run the following command to perform the configuration:
```console
./configure --prefix=/path/to/NETCDF CPPFLAGS="-I$hdf5/include -I$NETCDF/include" LDFLAGS="-LShdf5/lib -L$NETCDF/lib" CC=mpicc FC=mpif90 F77=mpif90
```
Step 5 Run the following commands to perform compilation and installation:
```console
make -j
```
make install

1.8.5 Obtaining the Source Code

Procedure

   Step 1  Download the FVCOM installation package FVCOM-4.1.tar.gz from the following address:
           http://fvcom.smast.umassd.edu/releases/fvcom-4.1.tar.gz

   Step 2  Use SFTP to upload the FVCOM installation package to the /path/to/FVCOM directory on the server.

1.8.6 Compiling and Installing FVCOM

Procedure

   Step 1  Use PuTTY to log in to the server as the root user.

   Step 2  Run the following command to create a main program installation directory:
           mkdir -p /path/to/FVCOM

   Step 3  Copy the installation package to the installation directory:
           cp fvcom-4.1.tar.gz /path/to/FVCOM

   Step 4  Run the following command to go to the main program installation directory:
           cd /path/to/FVCOM

   Step 5  Run the following command to decompress the installation package:
           tar -xvf fvcom-4.1.tar.gz

   Step 6  Perform the following operations to configure the files:

           1. Run the following command to copy the make.inc_example file and rename it as make.inc:
              cp Examples/Estuary/make.inc_example FVCOM_source/make.inc

           2. Run the following command to create a soft link for the make.inc file:
              ln -sf FVCOM_source/make.inc ./

           3. Run the following command to edit the make.inc file:
              a. vi make.inc

                 b. Press i to enter the editing mode and modify lines 51, 79, 80, 97, and 99.

                    Line 51:
                    TOPDIR = /path/to/FVCOM/FVCOM4.1/FVCOM_source

                    Lines 79 and 80:
                    LIBDIR = -L$(INSTALLDIR)/lib -L../METIS_source/metis -L./libs/julian
                    INCDIR = -I$(INSTALLDIR)/include -I../METIS_source/metis -I./libs/julian
Line 97:
IOLIBS = -L/path/to/NETCDF/lib -L/path/to/HDF5/lib -lnetcdff -lncdf -lhdf5.hl -lhdf5 -lz -lcurl -lm

Line 98:
IOINCS = -I/path/to/NETCDF/include -I/path/to/HDF5/include

c. Comment out and add contents as follows:
# Intel/MPI Compiler Definitions (SMAST)
#---------------------------------------------------------------------------
#       CPP      = /usr/bin/cpp
#       COMPILER = -DIFORT
#       CC       = mpicc
#       CXX      = mpicxx
#       CFLAGS   = -O3
#       FC       = mpiif90
#       DEBFLGS  = -check all traceback
# Use 'OPT = -O0 -g' for fast compile to test the make
# Use 'OPT = -xP' for fast run on em64t (Hydra and Guppy)
# Use 'OPT = -xN' for fast run on ia32 (Salmon and Minke)
# OPT      = -O0 -g
# OPT      = -axN -xN
# OPT      = -O3 -ffixed-line-length-none -ffree-form -ffree-line-length-none

# gfortran defs
#---------------------------------------------------------------------------
CPP      = /usr/bin/cpp
COMPILER = -DGFORTRAN
CC       = mpicc
CXX      = mpicxx
FC       = mpiif90
DEBFLGS  =
OPT      = -O3
LIB     =

d. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 7** Run the following command to create a compilation environment variable file:

1. Run the following command to create the environment variable file:
   a. vi env-fvcom.sh
   b. Press i to enter the editing mode and add the following content:

   ```
   export GCC_HOME=/path/to/GNU
   export PATH=$GCC_HOME/bin:$PATH
   export LD_LIBRARY_PATH=$GCC_HOME/lib:$GCC_HOME/lib64:$LD_LIBRARY_PATH
   export MPI_HOME=/path/to/OPENMPI
   export PATH=$MPI_HOME/bin:$PATH
   export LD_LIBRARY_PATH=$MPI_HOME/lib:$LD_LIBRARY_PATH
   export HDF5=/path/to/HDF5
   export PATH=$HDF5/bin:$PATH
   export LD_LIBRARY_PATH=$HDF5/lib:$LD_LIBRARY_PATH
   export NETCDF=/path/to/NETCDF
   export PATH=$NETCDF/bin:$PATH
   export LD_LIBRARY_PATH=$NETCDF/lib:$LD_LIBRARY_PATH
   ```
   c. Press Esc, enter :wq!, and press Enter to save the file and exit.

2. Run the following commands to load environment variables:
   ```
   source env-fvcom.sh
   ```

**Step 8** Run the following command to compile the METIS library:

1. Run the following command to go to the METIS installation directory:
   ```
   cd ./METIS_source
   ```
2. Run the following command to decompress the METIS installation package:
   ```
   tar -zxvf metis.tgz
   ```
3. Run the following command to copy the `metis-4.0.patch` file to the current directory:
   ```
   cp /path/to/METIS/metis-4.0.patch ./
   ```

4. Run the following commands to go to the `metis` directory and apply the patch file:
   ```
   cd metis
   patch -p2 < ../metis-4.0.patch
   ```

5. Run the following commands to create directories:
   ```
   mkdir -p /path/to/FVCOM/FVCOM4.1/FVCOM_source/libs/install/lib
   mkdir -p /path/to/FVCOM/FVCOM4.1/FVCOM_source/libs/install/include
   mkdir -p /path/to/FVCOM/FVCOM4.1/FVCOM_source/libs/install/bin
   ```

6. Run the following commands to compile and install the METIS library:
   ```
   make -j
   make install
   ```

**Step 9** Run the following commands to compile the julian library:

1. Run the following command to go to the julian installation directory:
   ```
   cd ../..../FVCOM_source/libs
   ```

2. Run the following command to decompress the `julian.tgz` package:
   ```
   tar -zxvf julian.tgz
   ```

3. Run the following command to switch to the directory generated after decompression:
   ```
   cd julian
   ```

4. Run the following commands to perform compilation and installation:
   ```
   make -j
   make install
   ```

**Step 10** Run the following command to compile and install the FVCOM main program:

1. Run the following command to go to the FVCOM installation:
   ```
   cd /path/to/FVCOM/FVCOM4.1/FVCOM_source
   ```

2. Run the following command to edit the `mod_newinp.F` file:
   ```
   a. vi mod_newinp.F
   b. Press i to enter the editing mode and add the following content below the first `contains` statement:
   ```
   !----------------------------------------
   ! functions
   !----------------------------------------
   contains
   Character( Len = 256 ) Function N_Fmt( c , n )
   Character( Len = * ) , Intent( IN ) :: c
   Integer , Intent( IN ) :: n
   integer :: i , j
   character( len = 16 ) :: cn
   i = index( c , '<' )
   j = index( c , '>' )
   write( cn , '(g0)' ) n
   N_Fmt = c(:i-1) // Trim(adjustL(cn)) // c(j+1:)
   End Function N_Fmt
   ```
c. Modify the `mod_newinp.F` file as follows:

Line 352:

Before the modification:
write(*,'(A20,<size>F10.4)trim(argname)//': ',fval(1:SIZE))

After the modification:
write(*,N_Fmt('(A20,<size>F10.4)',SIZE))trim(argname)//': ',fval(1:SIZE))

Line 421:

Before the modification:
write(*,'(A20,<size>I10)trim(argname)//': ',ival(1:SIZE))

After the modification:
write(*,N_Fmt('(A20,<size>I10)',SIZE))trim(argname)//': ',ival(1:SIZE))

Line 494:

Before the modification:
write(*,'(A20,<size>L10)trim(argname)//': ',cval(1:SIZE))

After the modification:
write(*,N_Fmt('(A20,<size>L10)',SIZE))trim(argname)//': ',cval(1:SIZE))

Line 567:

Before the modification:
write(*,'(A20,<size>A10)trim(argname)//': ',sval(1:SIZE))

After the modification:
write(*,N_Fmt('(A20,<size>A10)',SIZE))trim(argname)//': ',sval(1:SIZE))

d. Press Esc, enter .wq!, and press Enter to save the file and exit.

3. Run the following commands to modify the `mod_scal.F`, `internal_step.F`, `adv_t.F`, and `adv_s.F` files. Modify the statements starting with IF(BACKWARD_ADVECTION in the files. Change == to .eqv and /= to .neqv.

```
sed -i 's/\/=\./\neqv\./g' mod_scal.F
sed -i 's/\==\./\eqv\./g' internal_step.F
sed -i 's/\==\./\==\./g' adv_t.F
sed -i 's/\==\./\==\./g' adv_s.F
```

4. Run the following command to compile and install the main program:

```
make
```

After the installation is successful, you can run the ll command to view the generated executable file fvcom in the current directory.

----End

### 1.8.7 Running and Verifying FVCOM

#### 1.8.7.1 Running and Verifying the Estuary Case

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to create a test directory:

```
mkdir -p /path/to/FVCOM/test
```
Step 3 Switch to the test directory.

```
cd /path/to/FVCOM/test
```

Step 4 Run the following command to copy the test case to the current directory:

```
cp -r /path/to/FVCOM/FVCOM4.1/Examples/Estuary/ ./
```

Step 5 Run the following command to switch to the run directory:

```
cd ./Estuary/run
```

Step 6 Run the following command to edit the case input file:

1. `vi tst_run.nml`
2. Press `i` to enter the editing mode and add single quotation marks to `tst_grd.dat`:
   ```
   GRID_FILE = 'tst_grd.dat'
   ```
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

Step 7 Run the following command to run the test:

```
time mpirun -n 24 --allow-run-as-root --bind-to core /path/to/FVCOM/
FVCOM4.1/FVCOM_source/fvcom --casename=tst
```

Step 8 Run the following command to view the current directory. After the command is executed successfully, the `.nc` result file is generated.

```
ll
```

Figure 1-12 Result example

```
-rw-r--r-- 1 root root 48517284 Feb 5 12:07 tst_0001.nc
-rw------- 1 root root 10578 Feb 5 09:55 tst_run.nml
```

----End

1.8.7.2 Running and Verifying the Inlet Case

Prerequisites

FVCOM needs to compile an executable file based on the case. If the inlet case is used, replace the `make.inc` file with the `make.inc` file of the Intel case (see 1.8.6 Compiling and Installing FVCOM). Modify the `mod_cstms_vars.F` and `mod_cstms_vars.F` files in the `/path/to/FVCOM/FVCOM_source` directory. Other steps are the same as those in 1.8.6 Compiling and Installing FVCOM.

Step 1 Run the following command to modify the `mod_cstms_vars.F` file.

1. `vi mod_cstms_vars.F`
2. Press `i` to enter the insert mode and modify the file as follows:
   
   Before the modification:
   ```
   494       (/       "bo_sd50"                                     &
   529       (/        "mean grain diameter"                    &
   563       character(len=80), dimension(34) :: bot_units = (" m","kg&
   568       character(len=80), dimension(5) :: bed_snames =
   570       character(len=80), dimension(5) :: bed_units = ("m","days","-","-","N/m2")
   ```

   ```
   620       character(len=80), dimension(5) :: bed_lnames = ("bed layer thickness","bed layer age","bed layer porosity","bed layer bio-diffusivity","bed critical stress")/
   ```

   After the modification:
   ```
   620       character(len=80), dimension(5) :: bed_lnames = ("bed layer thickness","bed layer age","bed layer porosity","bed layer bio-diffusivity","bed critical stress")/
   ```
After the modification:

```fortran
494     (/character(80):: "bo_sd50" &
529     (/ character(80):: "mean grain diameter" &
563     character(len=80), dimension(34) :: bot_units = (/character(80):: "m", "kg" &
568     character(len=80), dimension(5) :: bed_snames = (/character(80):: "bed_thick", "bed_age", "bed_por", "bed_diff", "bed_btcr")/
569     character(len=80), dimension(5) :: bed_lnames = (/character(80):: "bed layer thickness", "bed layer age", "bed layer porosity", "bed layer bio-diffusivity", "bed critical stress")/
570     character(len=80), dimension(5) :: bed_units = (/character(80):: "m", "days", "-", "-", "N/m2")
```

3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

---

**Step 2** Modify the `mod_sed_cstms.F` file.

1. `vi mod_sed_cstms.F`
2. Press **i** to enter the insert mode and modify the file as follows:

Before the modification:

```fortran
5797     IF(BACKWARD_ADVECTION==.TRUE.)THEN
```

After the modification:

```fortran
5797     IF(BACKWARD_ADVECTION .EQV. .TRUE.)THEN
```

3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

---

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to create a test directory and go to the test directory:

```bash
mkdir -p /path/to/FVCOM/test
cd /path/to/FVCOM/test
```

**Step 3** Run the following command to copy the test case to the current directory:

```bash
cp -r /path/to/FVCOM/FVCOM4.1/Examples/cohesive_bed/tidal_inlet/ ./
```

**Step 4** Run the following command to switch to the **run** directory:

```bash
cd ./tidal_inlet/run
```

**Step 5** Run the following command to edit the case input file:

1. `vi inlet_run.nml`
2. Press **i** to enter the editing mode and add single quotation marks to **none, inlet_nesting.nc, and inlet_nesting.nc**.

```fortran
NCNEST_NODE_FILES_WAVE  = 'none'
NESTING_FILE_NAME       = 'inlet_nesting.nc'
NESTING_FILE_NAME_WAVE  = 'inlet_nesting_wave.nc'
```

3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

**Step 6** Run the following command to run the test:

```bash
time mpirun -np 96 --allow-run-as-root --bind-to core /path/to/FVCOM/FVCOM4.1/FVCOM_source/fvcom --casename=inlet
```

Example:
1.8.8 More Information

Official FVCOM website:

http://codfish.smast.umassd.edu/

1.9 MITGCM c67o Porting Guide (CentOS 7.6)

1.9.1 Introduction

MIT General Circulation Model (MITgcm) is a numerical model designed for study of the atmosphere, ocean, and climate. MITgcm's flexible non-hydrostatic formulation enables it to efficiently simulate fluid phenomena over a wide range of scales. Its adjoint capabilities enable it to be applied to sensitivity questions and to parameter and state estimation problems. By employing fluid equation isomorphisms, a single dynamical kernel can be used to simulate flow of both the atmosphere and ocean.

For more information about MITgcm, visit the official MITgcm website.

Language: C/Fortran 90

Brief description: MITgcm is a modeling software for simulating large-scale ocean circulation.

Open-source license: GPL 3.0

Recommended Version

MITgcm_c67o

1.9.2 Environment Requirements

Hardware Requirements

Table 1-43 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
Software Requirements

*Table 1-44* lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>MITGCM</td>
<td>c67o</td>
<td><a href="http://mitgcm.org/download/">http://mitgcm.org/download/</a></td>
</tr>
<tr>
<td>NetCDF-C</td>
<td>4.7.0</td>
<td><a href="https://www.unidata.ucar.edu/downloads/netcdf/">https://www.unidata.ucar.edu/downloads/netcdf/</a></td>
</tr>
<tr>
<td>NetCDF-Fortran</td>
<td>4.4.5</td>
<td><a href="https://www.unidata.ucar.edu/downloads/netcdf/">https://www.unidata.ucar.edu/downloads/netcdf/</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.1</td>
<td><a href="http://hdfgroup.org/HDF5/">http://hdfgroup.org/HDF5/</a></td>
</tr>
<tr>
<td>Test case</td>
<td>exp4</td>
<td>Provided by the software.</td>
</tr>
</tbody>
</table>

OS Requirements

*Table 1-45* lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

1.9.3 Paths for Software Porting

This chapter lists the software installation paths involved in the MITgcm software porting.

*Table 1-46* Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5</td>
<td>The installation paths listed in this table are only examples. Shared</td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------</td>
<td>--------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NetCDF-C</td>
<td>paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installation path of NetCDF-Fortran</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/MITGCM</td>
<td>Installation path of MITgcm</td>
<td></td>
</tr>
</tbody>
</table>

1.9.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install HDF5.</td>
<td>For details, see 1.9.4.1 Installing HDF5.</td>
</tr>
<tr>
<td>3</td>
<td>Install NetCDF-C.</td>
<td>For details, see 1.9.4.2 Installing NetCDF-C.</td>
</tr>
<tr>
<td>4</td>
<td>Install NetCDF-Fortran.</td>
<td>For details, see 1.9.4.3 Installing NetCDF-Fortran.</td>
</tr>
</tbody>
</table>

1.9.4.1 Installing HDF5

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.1.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:
cd hdf5-1.10.1

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/HDF5 --build=aarch64-linux --enable-parallel --enable-shared --enable-fortran
```

Step 5 Run the following commands to compile and install HDF5:

```
make -j
make install
```

Step 6 Configure HDF5 environment variables.

```
export LD_LIBRARY_PATH=/path/to/HDF5
export HDF5=/path/to/HDF5
```

1.9.4.2 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the NetCDF-C installation package.

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-c-4.7.0
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --enable-fortran --disable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-large-file-tests --enable-largefile CPPFLAGS="-I/path/to/HDF5/include " LDFLAGS="-L/path/to/HDF5/lib -Wl,-rpath=/path/to/HDF5/lib " CFLAGS="-L/path/to/HDF5/lib -I/path/to/HDF5/include"
```

Step 5 Run the following commands to compile and install NetCDF-C:

```
make -j
make install
```

Step 6 Configure NetCDF-C environment variables.

```
export PATH=/path/to/NETCDF/bin:SPATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:SLD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End
1.9.4.3 Installing NetCDF-Fortran

**Procedure**

**Step 1**  Use PuTTY to log in to the server as the root user.

**Step 2**  Run the following command to decompress the NetCDF-Fortran installation package:

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

**Step 3**  Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-fortran-4.4.5
```

**Step 4**  Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --disable-static --enable-shared --enable-parallel-tests --enable-large-file-tests --enable-largefile
```

**Step 5**  Run the following commands to compile and install NetCDF-Fortran:

```
make -j
make install
```

----End

1.9.5 Obtaining the Source Code

**Procedure**

**Step 1**  Download the MITgcm installation package MITgcm_c67o.tar.gz.

URL: [http://mitgcm.org/download/](http://mitgcm.org/download/)

**Step 2**  Use SFTP to upload the MITgcm installation package to the /path/to/MITGCM directory on the server.

----End

1.9.6 Compiling and Installing MITgcm

**Procedure**

**Step 1**  Use PuTTY to log in to the server as the root user.

**Step 2**  Run the following command to create a main program installation directory:

```
mkdir -p /path/to/MITGCM
```

**Step 3**  Copy the installation package to the installation directory.

```
cp MITgcm_c67o.tar.gz /path/to/MITGCM
```

**Step 4**  Run the following command to go to the main program installation directory:
Step 5 Run the following command to decompress the installation package:

tar -xvf MITgcm_c670.tar.gz

Step 6 Run the following command to switch to the directory generated after decompression:

cd MITgcm

Step 7 Perform the following operations to configure the files:

1. Run the following command to switch to the `build_options` directory:
   
   cd tools/build_options

2. Create and edit the `Linux_arm64_gfortran` file.
   
   vi linux_arm64_gfortran

3. Press i to enter the editing mode and add the following content:

   ```bash
   #!/bin/bash
   MPI='true'
   CC=mpicc
   FC=mpif77
   F90C=mpif90
   DEFINES='-DWORDLENGTH=4 -DNML_TERMINATOR -DALLOW_USE_MPI -DALWAYS_USE_MPI'
   EXTENDED_SRC_FLAG='-ffixed-line-length-132'
   F90FIXEDFORMAT='-ffixed-form'
   GET_FC_VERSION="--version"
   OMPFLAG='-fopenmp'
   NOOPTFLAGS='-O0 -g'
   NOOPTFILES=""
   CFLAGS='-O3 -march=armv8.2-a'
   FFLAGS="$FFLAGS -fconvert=big-endian -fimplicit-none"
   # CFLAGS="$CFLAGS -mccmodel=medium"
   # FFLAGS="$FFLAGS -mccmodel=medium"
   # might want to use ' -fdefault-real-8' for fizhi pkg:
   # FFLAGS="$FFLAGS -fdefault-real-8 -fdefault-double-8"
   if test "x$IEEE" = x ; then     #- with optimisation:
     # full optimisation
     FOPTIM='-O3 -funroll-loops -march=armv8.2-a'
     NOOPTFILES="$NOOPTFILES ini_masks_etc.F"
     # can use -O2 (safe optimisation) to avoid Pb with some gcc version of -O3:
     # FOPTIM='-O2 -funroll-loops'
   else
     # these may also be useful, but require specific gfortran versions:
     # -Wno-std-intrinsics for gfortran <= 4.3
     # -Winline functions for gfortran >= 4.4
     # -Wno-tabs for gfortran >= 4.3
     # -Wno-unused-dummy-argument for gfortran >= 4.6
     # FFLAGS="$FFLAGS -Wall -Wamr -Winteresting -Wsurprising -Wline-truncation"
     # - or simply:
     # FFLAGS="$FFLAGS -Wall"
     # - to get plenty of warnings: -Wall -Wextra (older form: -Wall -W) or:
     # FFLAGS="$FFLAGS -Wconversion -Wimplicit-interface -Wunused-labels"
   if test "x$DEVEL" = x ; then  #- no optimisation + IEEE:
     FOPTIM='-O0'
   else
     # development/check options:
     FOPTIM='-O0 -g -fbounds-check'
   FOPTIM="$FOPTIM -fppe-trap=invalid,zero,overflow -finit-real=inf"
   ```

Kunpeng BoostKit for HPC
Porting Guide

Issue 14 (2021-07-23)  Copyright © Huawei Technologies Co., Ltd.
4. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 8** Perform the following steps to compile and install algorithms, exp4 for example:

1. Run the following command to switch to the target directory:
   ```bash
cd /path/to/MITGCM/MITgcm/verification/exp4/code
   ```

2. Run the following command to rename `SIZE.h_mpi` to `SIZE.h` to replace the original `SIZE.h` file.
   ```bash
   mv SIZE.h_mpi SIZE.h
   ```

3. Run the following command to switch to the target directory:
   ```bash
cd /path/to/MITGCM/MITgcm/verification/exp4/build
   ```

4. Run the following command to generate the `makefile` file:
   ```bash
   ../../../tools/genmake2 -mods=../code -mpi -of=../../../tools/build_options/linux_arm64_gfortran
   ```

5. Run the following commands to perform compilation and installation:
   ```bash
   make depend
   make -j
   ```

   ----End

**1.9.7 Running and Verifying MITgcm**

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to switch to the test directory.
   ```bash
cd /path/to/MITGCM/MITgcm/verification/exp4/run
   ```

**Step 3** Run the following command to copy the test case to the current directory:
   ```bash
cp ../input/* .
   ```
Step 4 Run the following command to start a test:

```
ctime mpirun -n 4 --allow-run-as-root ./mitgcmuv
```

Step 5 Run the following command to view the STDOUT.0000 output file:

```
vi STDOUT.0000
```

If the information shown in the figure is displayed, the test is successful:

Figure 1-13 Result example

---End

1.9.8 More Information

Official MITgcm website:

http://mitgcm.org/

MITgcm Installation Guide:

https://mitgcm.readthedocs.io/en/latest/overview/overview.html

1.10 POM POM2K Porting Guide (CentOS 7.6)

1.10.1 Introduction

Princeton Ocean Model (POM) is a three-dimensional shallow sea model of the inclined continental shelf vertically using the sigma coordinate. Based on the original equation set, the POM contains a second-order turbulence sub-model, which can provide the turbulent eddy coefficient that varies with the flow rate. Developed by Blumberg and Mellor in the 1970s, it has been improved with the joint efforts of many scholars. It can be used in estuaries, coastal areas and open oceans.
The main feature of POM is that it uses the sigma coordinate in the vertical direction, so it can be easily introduced into the continental shelf. In determining the vertical turbulence exchange coefficient, the second-order turbulence kinetic energy (TKE) closure scheme is used. Therefore, the vertical difference is divided into implicit ones, and the limitation of time on the vertical coordinate is eliminated, so that the resolution of the upper and lower boundary layers of the ocean is improved and the calculation is stable. To save the calculation time and increase the calculation stability of the model, the POM separates the positive pressure model and the oblique pressure model of the current by using different time steps. The outer model is two-dimensional with short time step. The inner model is three-dimensional with long time step. In the horizontal direction, the POM uses the Arakawa C interlaced grid system. In the vertical direction, the turbulence kinetic energy equation, the vertical diffusion exchange coefficient and the vertical velocity ω are staggered with the temperature, salinity and the current field. The horizontal coordinate system may be optional, and may be a curve orthogonal coordinate or a longitude and latitude coordinate.

For details about AmberTools, visit the official POM website.

Language: Fortran

Brief description: POM is the Princeton Ocean Model.

Open-source license: GPL 3.0

**Recommended Versions**

POM2K

**1.10.2 Environment Requirements**

**Hardware Requirements**

*Table 1-48* lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

**Software Requirements**

*Table 1-49* lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>POM</td>
<td>POM2K</td>
<td><a href="http://www.ccpo.odu.edu/POMWEB/userinfo.htm">http://www.ccpo.odu.edu/POMWEB/userinfo.htm</a></td>
</tr>
</tbody>
</table>
### Test case

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test case</td>
<td>GRID-DATA</td>
<td><a href="http://www.ccpo.odu.edu/POMWEB/userinfo.htm">http://www.ccpo.odu.edu/POMWEB/userinfo.htm</a></td>
</tr>
</tbody>
</table>

### OS Requirements

*Table 1-50* lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 1.10.3 Planning the Paths for Software Porting

*Table 1-51* lists the software installation paths involved in the POM software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see <a href="#">Planning the Installation Paths</a> in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/POM</td>
<td>Installation path of POM</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NETCDF</td>
<td></td>
</tr>
</tbody>
</table>

### 1.10.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.
Procedure

Table 1-52 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Environment for the Cluster Scenario&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing NetCDF-C</td>
<td>For details, see 1.10.4.1 Installing NetCDF-C.</td>
</tr>
</tbody>
</table>

1.10.4.1 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the NetCDF-C installation package:

```
tar -zxvf netcdf-c-4.4.1.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-c-4.4.1.1
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --enable-fortran --disable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-large-file-tests --enable-largefile
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

Step 6 Run the following commands to set the NetCDF-C environment variables:

```
export PATH=/path/to/NETCDF/bin:SPATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:SLD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

----End
1.10.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to download the POM and test case GRID-DATA:

```
wget -r -p -k -np -nc -e robots=off http://www.ccpo.odu.edu/POMWEB
zip -r POM.zip www.ccpo.odu.edu
```

Step 3 After the download is complete, use the SFTP tool to upload POM.zip to the /path/to/POM directory on the server.

----End

1.10.6 Generating the IC Data File

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to create a main program installation directory:

```
mkdir -p /path/to/POM
```

Step 3 Copy the installation package to the installation directory.

```
cp POM.zip /path/to/POM
```

Step 4 Run the following command to go to the main program installation directory:

```
cd /path/to/POM
```

Step 5 Run the following command to decompress the installation package:

```
unzip POM.zip
```

Step 6 Run the following commands to move the POM2K and GRID-DATA directories to the installation directory of the main program:

```
cd www.ccpo.odu.edu/POMWEB/
mv POM2K ../../../
mv GRID-DATA ../../../
```

Step 7 Run the following command to switch to the directory where the GRID-DATA test case is stored:

```
cd /path/to/POM/GRID-DATA
```

Step 8 Run the following commands to create the IC and include directories:

```
mkdir IC
mkdir -p /path/to/POM/GRID-DATA/IC/include
```

Step 9 Copy the header files gridborder, gridcom, and params to the include directory.
Step 10  Run the following command to edit the rungrid file:

1. **vi rungrid**

2. Press `i` to enter the insert mode and enter the following information:

   **Before the modification:**
   ```
   echo ' Start Compiling ...'
   g77 -O3 GRID.f
   #pgf77 GRID.f
   a.out > printout
   #
   # -------------- output files for POM2K ------------------------
   /bin/mv fort.40    IC.dat         # formatted IC,grid & constant wind
   # -------------- output files for MATLAB plotting ------------
   /bin/mv fort.43   ijk.dat         # IM, JM, KB, Z
   /bin/mv fort.44   plt.dat         # formatted grid
   /bin/mv fort.45   bnd.dat         # boundary points (gridborder)
   /bin/mv fort.46   wnd.dat         # wind velocity
   /bin/mv fort.47   dxy.dat         # dx,dx (km)
   #
   /bin/rm a.out
   /bin/rm fort.*
   echo ' *** END GRID ***'
   ```

   **After the modification:**
   ```
   echo ' Start Compiling ...'
   gfortran -O3 GRID.f -I/path/to/POM/GRID-DATA/IC/include
   #pgf77 GRID.f
   .a.out > printout
   #
   # -------------- output files for POM2K ------------------------
   mv fort.40    IC.dat         # formatted IC,grid & constant wind
   # -------------- output files for MATLAB plotting ------------
   mv fort.43   ijk.dat         # IM, JM, KB, Z
   mv fort.44   plt.dat         # formatted grid
   mv fort.45   bnd.dat         # boundary points (gridborder)
   mv fort.46   wnd.dat         # wind velocity
   mv fort.47   dxy.dat         # dx,dx (km)
   #
   #!/bin/rm a.out
   #!/bin/rm fort.*
   echo ' *** END GRID ***'
   ```

3. Press **Esc**, enter `:wq!`, and press **Enter** to save the file and exit.

Step 11  Run the following commands to copy data files to the IC directory:

```
cp /path/to/POM/GRID-DATA/TOPO.dat /path/to/POM/GRID-DATA/IC

cp /path/to/POM/GRID-DATA/TS.dat /path/to/POM/GRID-DATA/IC

cp /path/to/POM/GRID-DATA/WIND.dat /path/to/POM/GRID-DATA/IC

cp /path/to/POM/GRID-DATA/GRID.f /path/to/POM/GRID-DATA/IC

cp /path/to/POM/GRID-DATA/IC.dat /path/to/POM/GRID-DATA/IC
```

Step 12  Run the following rungrid commands to grant the permissions and execute it:

```
chmod 755 rungrid
./rungrid
```
The downloaded installation package contains the IC.dat file. To help you understand the file generation process, this section describes how to use the required file to generate the IC.dat file again.

----End

1.10.7 POM2K Model Running

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to switch to the POM installation directory:

```bash
cd /path/to/POM/POM
```

Step 3  Run the following command to decompress the POM2K installation package:

```bash
unzip POM2K.zip
```

Step 4  Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd POM2K
```

Step 5  Run the following commands to modify the pom2k.f file:

1. `vi pom2k.f`
2. Press i to enter the editing mode and change mode 1 to mode 3.

   Before the modification:
   ```
   iproblem=1
   ```

   After the modification:
   ```
   iproblem=3
   ```
3. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 6  Run the following commands to copy the header file gridcom in the installation package and the IC.dat file generated in 1.10.6 Generating the IC Data File to the current directory:

```bash
cp /path/to/POM/GRID-DATA/IC/include/gridcom ./
cp /path/to/POM/GRID-DATA/IC/IC.dat ./
```

Step 7  Run the following commands to modify the runpom2k file:

1. `vi runpom2k`
2. Press i to enter the editing mode and change mode 1 to mode 3.

   Before the modification:
   ```
   # Runscript for pom2k with example of reading IC from file.
   # IC file generated by /GRID-DATA/GRID.f     (T.E. Dec04)
   # Note: if "grid" not created here you can put it in pom2k.c
   # "params" replaces default values set in pom2k.f
   #
   # ------------------ PARAMETER FILES FOR EACH CASE --------------
   ```

```
```
# -- SEAMOUNT TEST CASE (initial cond. calc. in pom2k.f)
# echo ' parameter(im=65, jm=49, kb=21)' > grid
# echo ' iproblem= 1 ' > params
# echo ' days= 0.025 ' >> params
# echo ' prtd1= 0.0125 ' >> params
# echo ' dte= 6. ' >> params
#
# -- COAST TEST CASE (initial cond. from file)
# #echo ' parameter(im=41, jm=61, kb=16)' > grid
# #echo ' iproblem= 3 ' > params
# #echo ' days= 0.50 ' >> params
# #echo ' prtd1= 0.25 ' >> params
# #echo ' dte= 12. ' >> params
# cp IC.dat fort.40 # Initial Condition file
#
# ------------------ COMPILE & RUN ----------------------------
#
# g77 -O3 pom2k.f /usr/local/lib/libnetcdf.a
# ---- now netCDF output not used (see "CDF" in pom2k.f)
# gfortran -O3 pom2k.f
# f90 -O3 pom2k.f # or any other compiler you wish to use
#
# ------------------ OUTPUT FILES -----------------------------
#
# a.out > pom2k.out # printout file
#
# pom2k.nc # netCDF file for MATLAB plotting

After the modification:
#
# Runs for pom2k with example of reading IC from file.
# IC file generated by /GRID-DATA/GRID.f (T.E. Dec04)
# 
# Note: if "grid" not created here you can put it in pom2k.c
# "params" replaces default values set in pom2k.f
#
# # ------------------ PARAMETER FILES FOR EACH CASE -----------
# # -- SEAMOUNT TEST CASE (initial cond. calc. in pom2k.f)
# # # echo ' parameter(im=65, jm=49, kb=21)' > grid
# # echo ' iproblem= 1 ' > params
# # echo ' days= 0.025 ' >> params
# # echo ' prtd1= 0.0125 ' >> params
# # echo ' dte= 6. ' >> params
# #
# # -- COAST TEST CASE (initial cond. from file)
# # echo ' parameter(im=41, jm=61, kb=16)' > grid
# echo ' iproblem= 3 ' > params
# echo ' days= 0.50 ' >> params
# echo ' prtd1= 0.25 ' >> params
# echo ' dte= 12. ' >> params
# cp IC.dat fort.40 # Initial Condition file
#
# # ------------------ COMPILE & RUN ----------------------------
#
# g77 -O3 pom2k.f /usr/local/lib/libnetcdf.a
# ---- now netCDF output not used (see "CDF" in pom2k.f)
# gfortran -O3 pom2k.f
# f90 -O3 pom2k.f # or any other compiler you wish to use
#
# # ------------------ OUTPUT FILES -----------------------------
#
#/a.out > pom2k.out # printout file
Step 8 Run the following commands to grant the permissions to `runpom2k` and execute it:

```bash
chmod 755 runpom2k
./runpom2k
```

The following is an example of the command output.

**Figure 1-14 Result example**

![Command output example](image)

**NOTE**
After the command is executed, the `fort.71` file is generated in the current directory. This file is the required NC file.

--- End

1.10.8 More Information

Official POM website:

[http://www.ccppo.odu.edu/POMWEB/userinfo.htm](http://www.ccppo.odu.edu/POMWEB/userinfo.htm)

1.11 MOM 6 Porting Guide (CentOS 7.6)

1.11.1 Introduction

The Modular Ocean Model (MOM) is a three-dimensional ocean circulation model used to study the ocean climate system.

For more information about MOM, visit the official MOM website.

Programming language: C/Fortran

Brief description: modular ocean model

Open-source license: GPL 3.0

Recommended Version

MOM 6

1.11.2 Environment Requirements

Hardware Requirements

*Table 1-53* lists the hardware requirements.
Table 1-53 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 1-54 lists the software requirements.

Table 1-54 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOM</td>
<td>6</td>
<td><a href="https://github.com/NOAA-GFDL/MOM6-examples.git">https://github.com/NOAA-GFDL/MOM6-examples.git</a></td>
</tr>
<tr>
<td>NetCDF-C</td>
<td>4.7.0</td>
<td><a href="https://www.unidata.ucar.edu/downloads/netcdf/index.jsp">https://www.unidata.ucar.edu/downloads/netcdf/index.jsp</a></td>
</tr>
<tr>
<td>NetCDF-Fortran</td>
<td>4.4.5</td>
<td><a href="https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.5">https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.5</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.1</td>
<td><a href="http://hdfgroup.org/HDF/">http://hdfgroup.org/HDF/</a></td>
</tr>
<tr>
<td>Test case</td>
<td>double_gyre</td>
<td>Provided by the software.</td>
</tr>
</tbody>
</table>

OS Requirements

Table 1-55 lists the OS requirements.

Table 1-55 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

1.11.3 Planning the Paths for Software Porting

Table 1-56 lists the software installation paths involved in the MOM software porting.
### Table 1-56 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package for setting up the basic environment</td>
<td>For details, see section Planning the Installation Paths in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/PNETCDF</td>
<td>Installation path of PnetCDF</td>
<td>The installation paths listed in this table are examples only. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the porting process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NetCDF-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installation path of NetCDF-Fortran</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/MOM</td>
<td>Installation path of MOM</td>
<td></td>
</tr>
</tbody>
</table>

### 1.11.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages have been uploaded to the server using an SFTP tool.

**Configuration Process**

### Table 1-57 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>“Setting Up the Cluster Environment” in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Install PnetCDF.</td>
<td>1.11.4.1 Installing PnetCDF</td>
</tr>
<tr>
<td>3</td>
<td>Install HDF5.</td>
<td>1.11.4.2 Installing HDF5</td>
</tr>
<tr>
<td>4</td>
<td>Install NetCDF-C.</td>
<td>1.11.4.3 Installing NetCDF-C</td>
</tr>
<tr>
<td>5</td>
<td>Install NetCDF-Fortran.</td>
<td>1.11.4.4 Installing NetCDF-Fortran</td>
</tr>
</tbody>
</table>
1.11.4.1 Installing PnetCDF

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Decompress the PnetCDF installation package.

```
tar -xvf parallel-netcdf-1.9.0.tar.gz
```

**Step 3** Switch to the folder generated after decompression.

```
 cd parallel-netcdf-1.9.0
```

**Step 4** Run the following command to perform the configuration:

```
./configure --prefix=/path/to/PNETCDF --build=aarch64-linux --enable-shared --enable-fortran --enable-large-file-test
```

**Step 5** Run the following commands to compile and install PnetCDF:

```
make -j
make install
```

**Step 6** Configure PnetCDF environment variables.

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:SLD_LIBRARY_PATH
```

----End

1.11.4.2 Installing HDF5

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Decompress the HDF5 installation package.

```
tar -xvf hdf5-1.10.1.tar.gz
```

**Step 3** Switch to the folder generated after decompression.

```
 cd hdf5-1.10.1
```

**Step 4** Run the following command to perform the configuration:

```
./configure --prefix=/path/to/HDF5 --build=aarch64-linux --enable-parallel --enable-shared --enable-fortran
```

**Step 5** Run the following commands to compile and install HDF5:

```
make -j
make install
```

**Step 6** Configure HDF5 environment variables.
export LD_LIBRARY_PATH=/path/to/HDF5/lib:SDL_LIBRARY_PATH

----End

1.11.4.3 Installing NetCDF-C

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Decompress the NetCDF-C installation package.

tar -zxvf netcdf-c-4.7.0.tar.gz

**Step 3** Switch to the folder generated after decompression.

cd netcdf-c-4.7.0

**Step 4** Run the following command to perform the configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --enable-fortran --disable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-large-file-tests --enable-largefile
```

**Step 5** Run the following commands to compile and install NetCDF-C:

```
make -j
make install
```

**Step 6** Configure NetCDF-C environment variables.

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:SDL_LIBRARY_PATH
```

----End

1.11.4.4 Installing NetCDF-Fortran

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Decompress the NetCDF-Fortran installation package.

tar -zxvf netcdf-fortran-4.4.5.tar.gz

**Step 3** Switch to the folder generated after decompression.

cd netcdf-fortran-4.4.5

**Step 4** Run the following command to perform the configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --disable-static --enable-shared --enable-parallel-tests --enable-large-file-tests --enable-largefile
```

**Step 5** Run the following commands to compile and install NetCDF-Fortran:
make -j
make install
----End

1.11.5 Obtaining the Source Code

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following command to switch to the MOM installation directory:
   cd /path/to/MOM
Step 3 After verifying that the server can connect to the Internet, run the following commands to obtain the source code:
   git clone https://github.com/NOAA-GFDL/MOM6-examples.git MOM6-examples
   cd MOM6-examples
   git submodule init
   git submodule update --recursive
Step 4 Run the following commands to update the modules:
   git submodule update src/FMS
   git submodule update --init --recursive src/MOM6
   git submodule update src/SIS2
   git submodule update tools/matlab/gtools
   git submodule update tools/python/MIDAS
   cd src/MOM6
   git submodule init
   git submodule update
----End

1.11.6 Compiling and Installing MOM

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following commands to modify the linux-gnu.mk file:
   cd /path/to/MOM/MOM6-examples/src/mkmf/templates/
   sed -ri 's/mpich2/ompi/g' linux-gnu.mk
Step 3  Compile libfms.a.

```
mkdir -p /path/to/MOM/MOM6-examples/build/gnu/shared/repro/

cd /path/to/MOM/MOM6-examples/build/gnu/shared/repro/

../../../../../src/mkmf/bin/list_paths -l ../../../../../src/FMS

../../../../../src/mkmf/bin/mkmf -t ../../../../../src/mkmf/templates/linux-gnu.mk -p
libfms.a -c "-Duse_libMPI -Duse_netCDF -DSPMD" path_names

make NETCDF=3 REPRO=1 libfms.a -j
```

Step 4  Compile MOM 6.

```
cd /path/to/MOM/MOM6-examples

mkdir -p build/gnu/ocean_only/repro

cd build/gnu/ocean_only/repro/

../../../../src/mkmf/bin/list_paths -l ./../../../../src/MOM6/{config_src/infra/FMS1,config_src/memory/dynamic_symmetric,config_src/drivers/solo_driver,config_src/external,src/{*,*/*}}/../../../../src/mkmf/bin/mkmf -t ../../../../src/mkmf/templates/linux-gnu.mk -o '-l../../shared/repro' -p 'MOM6 -L../../shared/repro -lfms' -c "-Duse_libMPI -DSPMD" path_names

make NETCDF=3 REPRO=1 MOM6 -j
```

117 Running and Verifying MOM

**Procedure**

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to start a benchmark test:

```
cd /path/to/MOM/MOM6-examples/ocean_only/benchmark/
```

Step 3  Run the following commands to modify the case scale:

```
cat << EOF >> MOM_override
#override NIGLOBAL = 720
#override NJGLOBAL = 360
EOF
```

Run the following commands to modify the case time:

```
cat << EOF >> input.nml
&ocean_solo_nml
months = 0
days = 20 /
EOF
```

Step 4  Run the following command to create a RESTART directory:

```
mkdir RESTART
```

Step 5  Run the following command to start a benchmark test:
```bash
{ time -p mpirun --allow-run-as-root -np 128 -mca btl ^openib ../../build/gnu/ocean_only/repro/MOM6 > mom.log ;} 2>>mom.log
```

**Step 6** View the value of `real` in the `mom.log` log. A smaller value indicates a higher performance.

*Figure 1* shows an example output.

**Figure 1-15** An example result

<table>
<thead>
<tr>
<th>Ocean tracer halo updates</th>
<th>0.183541</th>
<th>0.929249</th>
<th>7.049786</th>
<th>0.781172</th>
<th>0.215</th>
<th>0</th>
<th>127</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ocean tracer global sync</td>
<td>0.834516</td>
<td>0.015657</td>
<td>0.049905</td>
<td>0.001080</td>
<td>0.000</td>
<td>41</td>
<td>0</td>
</tr>
<tr>
<td>Ocean diffusor tracer</td>
<td>2.243899</td>
<td>3.781352</td>
<td>2.497076</td>
<td>0.082122</td>
<td>0.804</td>
<td>33</td>
<td>0</td>
</tr>
<tr>
<td>Ocean epicycral diffusor tracer</td>
<td>1.055029</td>
<td>7.209221</td>
<td>5.116466</td>
<td>2.091056</td>
<td>0.010</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>Ocean surface forcing</td>
<td>0.232695</td>
<td>0.058809</td>
<td>0.295893</td>
<td>0.048976</td>
<td>0.001</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>Ocean forcing diagnostics</td>
<td>0.835005</td>
<td>0.030476</td>
<td>0.031742</td>
<td>0.000046</td>
<td>0.000</td>
<td>41</td>
<td>0</td>
</tr>
</tbody>
</table>

---

**1.11.8 Troubleshooting**

**An Error Is Reported During libfms.a Compilation**

**Symptom**

When `libfms.a` is compiled, an error message is reported stating "Package mpich2 was not found in the pkg-config search path."

**Possible Causes**

The `mpich2` parameter in `linux-gnu.mk` is not modified.

**Procedure**

Check and modify the `linux-gnu.mk` file. For details, see *Step 2*.

**An Error Is Reported During libfms.a Compilation**

**Symptom**

When `libfms.a` is compiled, an error message is reported stating "make: nc-config: Command not found."

**Possible Causes**

The NetCDF environment is incorrect.

**Procedure**

Check the NetCDF environment configuration or reinstall NetCDF-C (see *1.11.4.3 Installing NetCDF-C*).

**1.11.9 More Information**

MOM installation guide:

1.12 SMOKE 4.7 Porting Guide (CentOS 7.6)

1.12.1 Introduction

The Sparse Matrix Operator Kernel Emissions (SMOKE) modeling system is primarily an emissions processing system designed to create gridded, speciated, hourly emissions for input into a variety of air quality models, such as CMAQ, REMSAD, CAMX, and UAM. SMOKE supports area, biogenic, mobile (both onroad and nonroad), and point source emissions processing for criteria, particulate, and toxic pollutants. For biogenic emissions modeling, SMOKE uses the Biogenic Emission Inventory System, version 2.5 (BEIS2) and version 3.09 and 3.14 (BEIS3). SMOKE is also integrated with the onroad emissions model MOBILE6 and MOVES.

For more information about SMOKE, visit the official SMOKE website.

Programming language: Fortran 90

Brief description: source emission tool for air modeling using sparse matrix

Open-source license: LGPL-3.0

Recommended Version

SMOKE 4.7

1.12.2 Environment Requirements

Hardware Requirements

Table 1-58 lists the hardware requirements.

Table 1-58 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 1-59 lists the software requirements.

Table 1-59 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMOKE</td>
<td>smoke_v47.Linux2_x86_64ifort.tar.gz smoke_v47.nctox.data.tar.gz smoke_install_v47.csh</td>
<td><a href="https://github.com/CEMPD/SMOKE/releases">https://github.com/CEMPD/SMOKE/releases</a></td>
</tr>
<tr>
<td>Item</td>
<td>Version</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>I/O API</td>
<td>3.2</td>
<td><a href="https://github.com/cjcoats/ioapi-3.2">https://github.com/cjcoats/ioapi-3.2</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.1</td>
<td><a href="http://hdfgroup.org/HDF5/">http://hdfgroup.org/HDF5/</a></td>
</tr>
<tr>
<td>PNETCDF</td>
<td>1.8.0</td>
<td><a href="http://cucis.ece.northwestern.edu/projects/PnetCDF">http://cucis.ece.northwestern.edu/projects/PnetCDF</a></td>
</tr>
<tr>
<td>NETCDF</td>
<td>4.4.1.1</td>
<td><a href="https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1">https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1</a></td>
</tr>
<tr>
<td>NETCDF-F</td>
<td>4.4.1</td>
<td><a href="https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1">https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1</a></td>
</tr>
<tr>
<td>Test computing instance</td>
<td>$SMK_HOME/data/ge_dat</td>
<td>Provided by the software.</td>
</tr>
</tbody>
</table>

### OS Requirements

Table 1-60 lists the OS requirements.

**Table 1-60** OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 1.12.3 Planning the Paths for Software Porting

Table 1-61 lists the software installation paths involved in the SMOKE software porting.
Table 1-61 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see section Planning Data in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5</td>
<td>The installation path provided here is only an example. A shared path is recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/PNETCDF</td>
<td>Installation path of PnetCDF</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NetCDF</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/NETCDF-FORTRAN</td>
<td>Installation path of NetCDF-Fortran</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/SMOKE</td>
<td>Installation path of SMOKE</td>
<td></td>
</tr>
</tbody>
</table>

1.12.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Procedure

Table 1-62 Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>For details, see &quot;Setting Up the Cluster Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install HDF5.</td>
<td>See 1.12.4.1 Installing HDF5.</td>
</tr>
<tr>
<td>3</td>
<td>Install PnetCDF.</td>
<td>See 1.12.4.2 Installing PnetCDF.</td>
</tr>
<tr>
<td>4</td>
<td>Install NetCDF.</td>
<td>See 1.12.4.3 Installing NetCDF.</td>
</tr>
<tr>
<td>5</td>
<td>Install NetCDF-Fortran.</td>
<td>See 1.12.4.4 Installing NetCDF-Fortran.</td>
</tr>
</tbody>
</table>
1.12.4.1 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the HDF5 installation package.

```bash
tar zxvf hdf5-1.10.1.tar.gz
```

Step 3 Switch to the directory generated after the HDF5 installation package is decompressed.

```bash
cd hdf5-1.10.1
```

Step 4 Set environment variables before compilation.

```bash
export MPICC=/path/to/OPENMPI/bin/mpicc
export MPICXX=/path/to/OPENMPI/bin/mpicxx
export MPIFC=/path/to/OPENMPI/bin/mpifort
export CC=$MPICC
export CXX=$MPICXX
export FC=$MPIFC
export F77=$MPIFC
```

Step 5 Run the following command to perform the configuration:

```bash
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared --enable-fortran20
```

Step 6 Run the following commands to compile and install HDF5:

```bash
make -j 16
make install
```

Step 7 Load environment variables.

```bash
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:SLD_LIBRARY_PATH
```

--- End

1.12.4.2 Installing PnetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the PnetCDF installation package.

```bash
tar -xvf parallel-netcdf-1.8.0.tar.bz2
```
Step 3 Switch to the directory generated after the PnetCDF installation package is decompressed.

```bash
cd parallel-netcdf-1.8.0
```

Step 4 Run the following command to perform the configuration:

```bash
./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu
CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC"
```

Step 5 Run the following commands to compile and install PnetCDF.

```bash
make -j 16
make install
```

Step 6 Load environment variables.

```bash
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

1.12.4.3 Installing NetCDF

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF installation package.

```bash
tar -xvf netcdf-4.4.1.1.tar.gz
```

Step 3 Switch to the directory generated after the NetCDF installation package is decompressed.

```bash
cd netcdf-4.4.1.1
```

Step 4 Set environment variables before compilation.

```bash
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/PNETCDF/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -Wl,-rpath=/path/to/HDF5/lib"
export CFLAGS="-L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -I/path/to/HDF5/include"
```

Step 5 Run the following command to perform the configuration:

```bash
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile
```

Step 6 Run the following commands to compile and install NetCDF.

```bash
make -j 16
```
1.12.4.4 Installing NetCDF-Fortran

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the NetCDF-Fortran installation package.

```
tar -xvf netcdf-fortran-4.4.1.tar.gz
```

Step 3 Switch to the directory generated after the NetCDF-Fortran installation package is decompressed.

```
cd netcdf-fortran-4.4.1
```

Step 4 Set environment variables before compilation.

```
export CPPFLAGS="-I/path/to/HDF5/include -I/path/to/NETCDF/include"
export LDFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -Wl,-rpath=/path/to/HDF5/lib -Wl,-rpath=/path/to/NETCDF/lib"
export CFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
export CXXFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
export FCFLAGS="-L/path/to/HDF5/lib -L/path/to/NETCDF/lib -I/path/to/HDF5/include -I/path/to/NETCDF/include"
```

Step 5 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static
```

Step 6 Run the following commands to compile and install NetCDF-Fortran:

```
make -j 16
make install
```

Step 7 Run the following command to delete environment variables:

```
unset CC CXX FC F77 CPPFLAGS LDFLAGS CFLAGS CXXFLAGS FCFLAGS
```

1.12.5 Obtaining the Source Code

Procedure

Step 1 Download the SMOKE installation package.

Download address: https://github.com/CEMPD/SMOKE/releases
Step 2 Use SFTP to upload the SMOKE installation package to the `/path/to/SMOKE` directory on the server.

---End

1.12.6 Compiling and Installing SMOKE

1.12.6.1 Installing the I/O API

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to install SMOKE in csh:

```
chsh -s /bin/csh
```

**Step 3** Restart the system for the installation to take effect.

```
reboot
```

**Step 4** Set environment variables.

```
echo "setenv SMK_HOME /path/to/SMOKE" >> /root/.tcsh
echo "setenv BIN Linux2_aarch64gfort" >> /root/.tcsh
echo setenv PATH /path/to/OPENMPI/openmpi-4.0.1-gcc9.1/bin:/path/to/GNU/gcc9.1/bin:$PATH >> /root/.tcsh
echo setenv LD_LIBRARY_PATH /path/to/NETCDF/lib:$LD_LIBRARY_PATH >> /root/.tcsh
```

**Step 5** Run the following command to make the environment variables take effect:

```
source /root/.tcsh
```

**Step 6** Copy the installation packages to the installation paths:

```
cp smoke_v47.Linux2_x86_64ifort.tar.gz /path/to/SMOKE

cp smoke_v47.nctox.data.tar.gz /path/to/SMOKE

cp smoke_install.csh /path/to/SMOKE

cp ioapi-3.2.tar.gz /path/to/SMOKE

cd /path/to/SMOKE

source smoke_install.csh
```

**Step 7** Create the directory.

```
mkdir -p $SMK_HOME/subsys/ioapi

mkdir -p $SMK_HOME/subsys/ioapi/$BIN
```
Step 8  Decompress the ioapi-3.2.tar.gz installation package.

```bash
cp ioapi-3.2.tar.gz ./subsys/ioapi/
cd ./subsys/ioapi
tar -xvf ioapi-3.2.tar.gz
```

Step 9  Create the Makefile file.

```bash
cd $SMK_HOME/subsys/ioapi/ioapi
cp Makefile.nocpl Makefile
```

Step 10  Modify the Makefile file.

1. Run the `vi Makefile` command.
2. Press `i` to enter the insert mode and modify the file as follows:
   ```
   BASEDIR = ${SMK_HOME}/subsys/ioapi
   INSTDIR = $(BASEDIR)/${BIN}
   
   MFLAGS = -ffast-math -funroll-loops -march=armv8-a
   ```
3. Press Esc, enter `.wq!`, and press Enter to save the file and exit.

Step 11  Modify the Makeinclude.Linux2_x86_64gfort file.

```bash
cp Makeinclude.Linux2_x86_64gfort Makeinclude.SBIN
```

1. Run the `vi Makeinclude.Linux2_x86_64gfort` command.
2. Press `i` to enter the insert mode and modify the file as follows:
   ```
   MFLAGS = -ffast-math -funroll-loops -march=armv8-a
   ```
3. Press Esc, enter `.wq!`, and press Enter to save the file and exit.

Step 12  Run the following command to perform compilation:

```bash
make
```

Step 13  Run the following commands to set soft links:

```bash
cd ../$SBIN
ln -sf /path/to/NETCDF/lib/libnetcdf.so ./
lne -sf /path/to/NETCDF/lib/libnetcdff.so ./
```

Step 14  Run the following command to switch to the m3tools directory:

```bash
cd ../m3tools
```

Step 15  Modify the Makefile file.

```bash
cp Makefile.nocpl Makefile
```

1. Run the `vi Makefile` command.
2. Press `i` to enter the insert mode and modify the file as follows:
   ```
   BASEDIR = ${SMK_HOME}/subsys/ioapi
   INSTDIR = $(BASEDIR)/${BIN}
   ```
3. Press Esc, enter `.wq!`, and press Enter to save the file and exit.

Step 16  Run the following command to perform compilation:

```bash
make
```

----End
1.12.6.2 Installing SMOKE

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command:

   
   cd $SMK_HOME/subsys/smoke/assigns/

Step 3  Modify the configuration file.

   1. Run the vi ASSIGNS.nctox.cmaq.cb05_soa.us12-nc command.
   2. Press i to enter the insert mode and modify line 25 as follows:
      setenv BIN Linux2_aarch64gfort
   3. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 4  Modify the Makeinclude file.

   1. cd $SMK_HOME/subsys/smoke/src
   2. vi Makeinclude
   3. Press i to enter the insert mode and modify the file as follows:
      INSTDIR = $(OBJDIR)/$(BIN)
      #EFLAG = -extend-source 132 -zero
      EFLAG = -fixed-line-length-132 -fno-backslash
      IOLIB = -L$(IOBIN) -lioapi -lnetcdf -lnetcdff
   4. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 5  Modify the biog/normbeis361.f file.

   1. Run the vi biog/normbeis361.f command.
   2. Press i to enter the insert mode and change the types of the IS_AG and IS_TAG functions in line 161 from INTEGER to LOGICAL.
   3. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 6  Modify the smkinven/rdinvsrcs.f file.

   1. Run the vi smkinven/ rdinvsrcs.f command.
   2. Press i to enter the insert mode and change the type of the GETPID function in line 86 from EXTERNAL to INTRINSIC.
   3. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 7  Modify the emutil/gentpro.f file.

   1. Run the vi emutil/gentpro.f command.
   2. Press i to enter the insert mode and change the code in lines 1663 and 1695 as follows:
      1663  (( PROF_MON( S,NP )), NP = 1,12 )
      1695  (( PROF_DAY( S,NP ) ), NP = 1,31 )
   3. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 8  Create the directory.

   
   mkdir ${SMK_HOME}/subsys/smoke/${BIN}

Step 9  Run the following commands to complete compilation:
source /path/to/SMOKE/subsys/smoke/assigns/ASSIGNS.nctox.cmaq.cb05_soa.us12-nc
make

1.12.7 Running and Verifying SMOKE

Procedure

Step 1  Use PuTTY to log in to the server as the root user.
Step 2  Run the following command to perform parallel computing:
cd $SMK_HOME/subsys/smoke/script/run/
Step 3  Run the following command to test the execution time of each script:
time/Script name
Example: time./smk_point_nctox.csh

Figure 1-16 shows an example of the test result.

<table>
<thead>
<tr>
<th>script</th>
<th>time/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>smk_point_nctox.csh</td>
<td>36.469</td>
</tr>
<tr>
<td>smk_area_nctox.csh</td>
<td>10.859</td>
</tr>
<tr>
<td>smk_ratoperhour_nctox.csh</td>
<td>16.874</td>
</tr>
<tr>
<td>smk_ratoperdistance_nctox.csh</td>
<td>34.97</td>
</tr>
</tbody>
</table>

1.12.8 Troubleshooting

An Error Is Reported When SMOKE Is Run

Symptom

Possible causes
The code space of fitrxref.f:192 is too long.

Procedure

Step 1  Use PuTTY to log in to the server as the root user.
Step 2  Run the following command to switch to the directory in which the Makefileinclude file is stored:
cd $SMK_HOME/subsys/smoke/src

Step 3  Modify the Makefileinclude file.

1. Run vi Makefileinclude.
2. Press i to enter the insert mode and modify the file as follows:
   EFLAG = -ffixed-line-length-132 -fno-backslash
3. Press Esc, enter :wq!, and press Enter to save the file and exit.

1.12.9 More Information

Official installation guide:

https://www.cmascenter.org

1.13 SPECFEM3D GLOBE 7.0.0 Porting Guide (CentOS 7.6)

1.13.1 Introduction

SPECFEM3D GLOBE simulates global and regional (continental scale) seismic wave propagation. Additional new model routines are provided for the Comprehensive Earth Model (CEM) project, generic point-profile models (PPM) and Gauss-Lobatto-Legendre based models (GLL), with complementary tools for postprocessing adjoint sensitivity kernels and gradient-based model updates. The structure of the software has been simplified to facilitate easier implementation of new 3D models. The code accommodates general moment tensor files, and provides complete information in the SAC headers, as explained in detail in the updated user manual. Loop-vectorization helps reduce the total number of memory accesses performed in each spectral element and improve code vectorization, enhancing numerical performance of the version.

For more information about SPECFEM3D GLOBE, visit the official SPECFEM3D GLOBE website.

Language: C/Fortran

Brief description: SPECFEM3D GLOBE simulates global and regional (continental scale) seismic wave propagation.

Open-source protocol: GNU Public License

Recommended Version

SPECFEM3D GLOBE 7.0.0

1.13.2 Environment Requirements

Hardware Requirements

Table 1-63 lists the hardware requirements.
### Table 1-63 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

### Software Requirements

Table 1-64 lists the software requirements.

#### Table 1-64 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPECFEM3D GLOBE</td>
<td>7.0.0</td>
<td><a href="https://github.com/geodynamics/specfem3d_globe/archive/v7.0.0.tar.gz">https://github.com/geodynamics/specfem3d_globe/archive/v7.0.0.tar.gz</a></td>
</tr>
</tbody>
</table>

### OS Requirements

Table 1-65 lists the OS requirements.

#### Table 1-65 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 1.13.3 Planning the Paths for Software Porting

Table Paths for software porting lists the software installation paths involved in the SPECFEM3D GLOBE software porting.

#### Table 1-66 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/SPECFEM3D_GLOBE</td>
<td>Installation path of SPECFEM3D GLOBE</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

1.13.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Procedure

Table 1-67 Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
</tbody>
</table>

1.13.5 Obtaining the Source Code

Procedure

**Step 1** Download the SPECFEM3D GLOBE installation package v7.0.0.tar.gz.

URL: [https://github.com/geodynamics/specfem3d_globe/archive/v7.0.0.tar.gz](https://github.com/geodynamics/specfem3d_globe/archive/v7.0.0.tar.gz)

**Step 2** Use the SFTP tool to upload the SPECFEM3D GLOBE installation package to the /path/to/SPECFEM3D_GLOBE directory on the server.

----End

1.13.6 Compiling and Installing SPECFEM3D GLOBE

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.
Step 2 Run the following command to create a main program installation directory:

```
mkdir -p /path/to/SPECFEM3D_GLOBE
```

Step 3 Run the following commands to decompress the source code package and go to the source code directory:

```
cd /path/to/SPECFEM3D_GLOBE
tar -xvf v7.0.0.tar.gz
cd specfem3d_globe-7.0.0
```

Step 4 Run the following commands to load environment variables:

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
$LD_LIBRARY_PATH
```

Step 5 Run the following command to generate a configuration file:

```
./configure --enable-openmp FC=mpif90 MPIFC=mpif90 CC=mpicc
MPICC=mpicc
```

Step 6 Run the following command to modify the `Par_file` file:

1. `vi DATA/Par_file`
2. Press `i` to enter the insert mode and modify the lines 23 and 24 in the file.
   
<table>
<thead>
<tr>
<th>Before the modification:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPROC_XI</td>
</tr>
<tr>
<td>NPROC_ETA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>After the modification:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPROC_XI</td>
</tr>
<tr>
<td>NPROC_ETA</td>
</tr>
</tbody>
</table>

3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

Step 7 Run the following command to perform compilation.

```
make -j
```

1.13.7 Running and Verifying SPECFEM3D GLOBE

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Switch to the test directory.

```
cd /path/to/SPECFEM3D_GLOBE/specfem3d_globe-7.0.0
```

Step 3 Run the following commands to perform the test:

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
SLD_LIBRARY_PATH
mpirun --allow-run-as-root -np 64 --mca btl ^openib ./bin/xmeshfem3D
```
mpirun --allow-run-as-root -np 64 --mca btl ^openib ./bin/xspecfem3D
less OUTPUT_FILES/output_solver.txt

If the information shown in the following figure is displayed, the test is successful.

Figure 1-17 Result example

Time Loop Complete. Timing info:
Total elapsed time in seconds = 40.010946430999998
Total elapsed time in hh:mm:ss = 0 h 00 m 40 s

----End

1.14 CMAQ 5.3.1 Porting Guide (CentOS 7.6)

1.14.1 Introduction

The Community Multiscale Air Quality (CMAQ) modeling system is at the heart of the third-generation air quality modeling system (Models-3) developed by the U.S. Environmental Protections Agency (USEPA). It is a three-dimensional Euler grid-based atmospheric chemistry and transmission simulation system to provide sound estimates of ozone, particulates, toxics, and acid deposition. The CMAQ model comprehensively represents air quality problems at different spatial scales and is widely used to address environmental issues and to conduct atmospheric research.

For more information, visit the CMAQ official website.

Programming language: Fortran

Brief description: a three-dimensional Euler grid-based atmospheric chemistry and transmission simulation system

Recommended Version

CMAQ 5.3.1

1.14.2 Environment Requirements

Hardware Requirements

Table 1-68 lists the hardware requirements.

Table 1-68 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
Software Requirements

Table 1-69 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMAQ</td>
<td>5.3.1</td>
<td><a href="https://codeload.github.com/USEPA/CMAQ/tar.gz/CMAQv5.3.1_19Dec2019">https://codeload.github.com/USEPA/CMAQ/tar.gz/CMAQv5.3.1_19Dec2019</a></td>
</tr>
<tr>
<td>GNU</td>
<td>9.3.0</td>
<td><a href="https://codeload.github.com/gcc-mirror/gcc/tar.gz/releases/gcc-9.3.0">https://codeload.github.com/gcc-mirror/gcc/tar.gz/releases/gcc-9.3.0</a></td>
</tr>
<tr>
<td>HPC-X</td>
<td>2.6</td>
<td><a href="https://www.mellanox.com/page/hpcx_eula?mtag=hpc-x&amp;mrequest=downloads&amp;mtype=hpc&amp;mver=hpc-x&amp;mnname=v2.6/hpcx-v2.6.0-gcc-OFED-4.17-redhat7.6-aarch64.tbz">https://www.mellanox.com/page/hpcx_eula?mtag=hpc-x&amp;mrequest=downloads&amp;mtype=hpc&amp;mver=hpc-x&amp;mnname=v2.6/hpcx-v2.6.0-gcc-OFED-4.17-redhat7.6-aarch64.tbz</a></td>
</tr>
<tr>
<td>HDFS</td>
<td>1.10.1</td>
<td><a href="https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/">https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/</a></td>
</tr>
<tr>
<td>NetCDF-C</td>
<td>4.7.0</td>
<td><a href="https://github.com/Unidata/netcdf-c/releases/tag/v4.7.0">https://github.com/Unidata/netcdf-c/releases/tag/v4.7.0</a></td>
</tr>
<tr>
<td>NetCDF-Fortran</td>
<td>4.4.5</td>
<td><a href="https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.5">https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.5</a></td>
</tr>
<tr>
<td>OPTIMIZED-ROUTINES</td>
<td>V20.02</td>
<td><a href="https://codeload.github.com/ARM-software/optimized-routines/tar.gz/v20.02">https://codeload.github.com/ARM-software/optimized-routines/tar.gz/v20.02</a></td>
</tr>
<tr>
<td>ioapi</td>
<td>3.2</td>
<td><a href="https://codeload.github.com/cjcoats/ioapi-3.2/tar.gz/2020111">https://codeload.github.com/cjcoats/ioapi-3.2/tar.gz/2020111</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 1-70 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
1.14.3 Planning the Paths for Software Porting

This section describes the software installation paths involved in the CMAQ software porting.

Table 1-71 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5.</td>
<td>The installation paths listed in this table are only for reference. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NetCDF-C.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installation path of NetCDF-fortran.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/MATH</td>
<td>Installation path of OPTIMIZED ROUTINES.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/CMAQ</td>
<td>Installation path of CMAQ.</td>
<td></td>
</tr>
</tbody>
</table>

1.14.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 1-72 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install HDF5.</td>
<td>For details, see 1.14.4.1 Installing HDF5.</td>
</tr>
</tbody>
</table>
### 1.14.4.1 Installing HDF5

**Procedure**

**Step 1** Use PuTTY to log in to the server as the `root` user.

**Step 2** Run the following command to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.1.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:

```
cd hdf5-1.10.1
```

**Step 4** Run the following command to perform configuration:

```
./configure --prefix=/path/to/HDF5 --build=aarch64-linux --enable-parallel --enable-shared --enable-fortran
```

**Step 5** Run the following commands to perform compilation and installation:

```
make -j
make install
```

**Step 6** Run the following command to set the HDF5 environment variable:

```
export LD_LIBRARY_PATH=/path/to/HDF5/lib:SLD_LIBRARY_PATH
```

---End

### 1.14.4.2 Installing NetCDF-C

**Procedure**

**Step 1** Use PuTTY to log in to the server as the `root` user.

**Step 2** Run the following command to decompress the NetCDF-C installation package:

```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:

```
 cd netcdf-c-4.7.0
```
1.14.4.3 Installing NetCDF-Fortran

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the NetCDF-Fortran installation package:

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:

```
cd netcdf-fortran-4.4.5
```

**Step 4** Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --disable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-large-file-tests --enable-largefile
```

**Step 5** Run the following commands to perform compilation and installation:

```
make -j
make install
```

1.14.4.4 Installing OPTIMIZED-ROUTINES

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the OPTIMIZED-ROUTINES installation package:

```
tar -xvf optimized-routines-20.02.tar.gz
```
Step 3  Run the following command to switch to the directory generated after decompression:
   cd optimized-routines-20.02

Step 4  Run the following command to create the MATH directory:
   mkdir -p /path/to/MATH

Step 5  Run the following command to copy the configuration file:
   cp config.mk.dist config.mk

Step 6  Run the following command to perform compilation and installation:
   make -j16

Step 7  Run the following command to copy the related library file:
   cp -r ./build/* /path/to/MATH

Step 8  Run the following commands to add MATH to environment variables:
   export PATH=/path/to/MATH/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/MATH/lib:$LD_LIBRARY_PATH

---End

1.14.5 Obtaining the Source Code

Procedure

Step 1  Download the CMAQ dependency ioapi-3.2-2020111.tar.gz at https://codeload.github.com/cjcoats/ioapi-3.2/tar.gz/2020111

Step 2  Download the CMAQ installation package CMAQ-CMAQv5.3.1_19Dec2019.tar.gz at https://codeload.github.com/USEPA/CMAQ/tar.gz/CMAQv5.3.1_19Dec2019

Step 3  Use the SFTP tool to upload the CMAQ dependency and installation package to the /path/to/CMAQ directory.

----End

1.14.6 Compiling and Installing CMAQ

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to go to the CMAQ directory:
   cd /path/to/CMAQ

Step 3  Run the following commands to decompress the package and rename ioapi:
   tar -xvf ioapi-3.2-2020111.tar.gz
   mv ioapi-3.2-2020111 ioapi-3.2
Step 4 Run the following commands to copy the configuration file:

```
  cd ioapi-3.2
  cp ioapi/Makeinclude.Linux2_ia64gfort ioapi/Makeinclude.Linux4_aarch64
```

Step 5 Edit the configuration file.

1. Run the following command to modify the `Makeinclude.Linux4_aarch64` configuration file:

```
vim ioapi/Makeinclude.Linux4_aarch64
```

2. Press `i` to enter the editing mode.

Modify the compiler options:

```
CC   = mpicc
CXX  = mpicxx
FC   = mpifort
```

Comment out the line corresponding to the `FSFLAGS` keyword, for example:

```
#FSFLAGS = -save
```

3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

Step 6 Run the following commands to copy Makefile and configure HOME:

```
  cp ioapi/Makefile.nocpl ioapi/Makefile
  export HOME=/path/to/CMAQ
```

Step 7 Run the following command to copy the configuration file:

```
  cp m3tools/Makefile.nocpl m3tools/Makefile
```

Step 8 Edit the configuration file.

1. Run the following command to modify the `Makefile` configuration file:

```
vim m3tools/Makefile
```

2. Press `i` to enter the editing mode and modify the script as follows:

```
LIBS = -L$(OBJDIR) -liapi -L/path/to/NETCDF/lib -lnetcdff -lnetcdf -L/path/to/HDF5/lib -lhdf5_hl -lhdf5 -lz $(OMPLIBS) $(ARCHLIB) $(ARCHLIBS)
```

3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

Step 9 Run the following command to copy the configuration file:

```
  cp Makefile.template Makefile
```

Step 10 Edit the configuration file.

1. Run the following command to edit the configuration file:

```
vim Makefile
```

2. Press `i` to enter the editing mode.

Modify the following content and delete the comment tag:

```
BIN        = Linux4_aarch64
BASEDIR    = ${PWD}
INSTALL    = ${HOME}
LIBINST    = $(INSTALL)/$(BIN)
BININST    = $(INSTALL)/$(BIN)
CPLMODE    = nocpl
IOAPIDEFS  = "-DIOAPI_NCF4"

NCFLIBS    = -L/path/to/NETCDF/lib -lnetcdff -lnetcdf -L/path/to/HDF5/lib -lhdf5_hl -lhdf5 -lz
```

Modify the `NCFLIBS` item:

```
NCFLIBS    = -L/path/to/NETCDF/lib -lnetcdff -lnetcdf -L/path/to/HDF5/lib -lhdf5_hl -lhdf5 -lz
```
3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

**Step 11** Run the following command to compile ioapi:

```
make BIN=Linux4_aarch64
```

**Step 12** Modify the **STATE3.EXT** file.

1. Run the following command to modify the **STATE3.EXT** file:

   ```
   vim ioapi/STATE3.EXT
   ```

2. Press **i** to enter the editing mode. Delete **&** at the end of some lines in the **STATE3.EXT** file, as shown in the red box in the following figure:

   ![STATE3.EXT file](image)

3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

**Step 13** Run the following commands to decompress the package and go to the directory:

```
tar -xvf CMAQ-CMAQv5.3.1_19Dec2019.tar.gz
cd CMAQ-CMAQv5.3.1_19Dec2019
```

**Step 14** Edit the configuration file.

1. Run the following command to edit the configuration file:

   ```
vim bldit_project.csh
   ```

2. Press **i** to enter the editing mode and modify the script as follows:

   ```
   set CMAQ_HOME = /path/to/CMAQ/CMAQ_Project
   ```

3. Press **Esc**, enter **:wq!**, and press **Enter** to save the file and exit.

**Step 15** Run the following command to create the files required for initialization:

```
./bldit_project.csh
```

**Step 16** Run the following command to switch to the working directory:

```
cd ../CMAQ_Project/
```

**Step 17** Edit the configuration file.

1. Run the following command to edit the configuration file:

   ```
vim config_cmaq.csh
   ```

2. Press **i** to enter the editing mode.

   In the **case gcc** area, modify the paths to dependencies, as shown in the following figure:
Step 18 Run the following command to connect to various dependent libraries:

./config_cmaq.csh gcc 9.3.0

Step 19 Run the following commands in sequence to go to the compilation directory and compile the main program:

cd CCTM/scripts/

./bldit_cctm.csh gcc 9.3.0

After the compilation is complete, the executable program CCTM_v531.exe is generated in the BLD_CCTM_v531_gcc9.3.0 directory.

1.14.7 Running and Verifying CMAQ

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the case files:

tar -xvf CMAQv5.3.1_Benchmark_2Day_Input.tar.gz -C /path/to/CMAQ/CMAQ_Project/data

Step 3 Run the following command to switch to the test directory:

cd /path/to/CMAQ/CMAQ_Project/CCTM/scripts

Step 4 Configure the running script.

1. Run the following command to configure the running script:

vim run_cctm_Bench_2016_12SE1.csh

2. Press i to enter the editing mode.

   Change the value of compiler to gcc and the value of compilerVrsn to 9.3.0:

setenv compiler gcc
setenv compilerVrsn 9.3.0

   Change the values of NPCOL and NPROW. The product of the two values is equal to the number of parallel cores. For example:

@ NPCOL = 12; @ NPROW = 8

   Modify the INPDIR path, for example:

setenv INPDIR /path/to/CMAQ/CMAQ_Project/data/CMAQv5.3.1_Benchmark_2Day_Input/2016_12SE1

---End
Modify the OpenMPI running parameters. For example:

```
/usr/bin/time -p mpirun -np $NPROCS -x LD_PRELOAD=/path/to/MATH/lib/libmathlib.so --allow-run-as-root --mca btl ^openib $BLD/$EXEC
```

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 5** Run the following command to perform the test:

```
csh -x /path/to/CMAQ/CMAQ_Project/CCTM/scripts/run_cctm_Bench_2016_12SE1.csh 2>&1 | tee -a /path/to/CMAQ/CMAQ_Project/CCTM/scripts/csh.log
```

**Step 6** Check the value of **Total Time** in the `csh.log` file. The unit is second. A smaller value indicates better performance.

*Figure 1-18* shows an example of the running result.

---

### 1.14.8 Troubleshooting

#### Problem 1: An error is reported during CMAQ compilation.

**Symptom**

An error "Error:Syntax in COMMON statement at (1) STATE3.EXT:174:27:" is reported during CMAQ compilation.

**Possible Causes**

The data format is incorrect.

**Procedure**

Modify the `STATE3.EXT` file by referring to **Step 12**.

---

### 1.15 SWAN 41.31 Porting Guide (CentOS 7.6)
1.15.1 Introduction

SWAN is a third-generation wave model for obtaining realistic estimates of wave parameters in coastal areas, lakes and estuaries from given wind, bottom and current conditions. However, SWAN can be used on any scale relevant for wind-generated surface gravity waves. The model is based on the wave action balance equation with sources and sinks.

For more information about SWAN, visit the official SWAN website.

Programming language: Fortran

Brief description: a wave model

Recommended Version

SWAN 41.31

1.15.2 Environment Requirements

Hardware Requirements

Table 1-73 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 1-74 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open MPI</td>
<td>4.0.1</td>
<td><a href="https://www.open-mpi.org/software/ompi/v4.0/">https://www.open-mpi.org/software/ompi/v4.0/</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.5</td>
<td><a href="https://portal.hdfgroup.org/display/support/HDF5+1.10.5#files">https://portal.hdfgroup.org/display/support/HDF5+1.10.5#files</a></td>
</tr>
</tbody>
</table>
### OS Requirements

*Table 1-75* lists the OS requirements.

#### Table 1-75 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 1.15.3 Planning the Paths for Software Porting

*Table 1-76* lists the software installation paths involved in the SWAN software porting.

#### Table 1-76 Paths for software porting

<table>
<thead>
<tr>
<th>N o.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see “Planning the Installation Paths” in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5.</td>
<td>The installation paths listed in this table are only examples. Shared</td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------</td>
<td>------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/PNETCDF</td>
<td>Installation path of PnetCDF.</td>
<td>paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>4</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NetCDF-C.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installation path of NetCDF-Fortran.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/SWAN</td>
<td>Installation path of SWAN.</td>
<td></td>
</tr>
</tbody>
</table>

### 1.15.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Configuration Process

**Table 1-77 Configuration process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Installing HDF5.</td>
<td>For details, see 1.15.4.1 Installing HDF5.</td>
</tr>
<tr>
<td>3</td>
<td>Installing PnetCDF.</td>
<td>For details, see 1.15.4.2 Installing PnetCDF.</td>
</tr>
<tr>
<td>4</td>
<td>Installing NetCDF-C.</td>
<td>For details, see 1.15.4.3 Installing NetCDF-C.</td>
</tr>
<tr>
<td>5</td>
<td>Installing NetCDF-Fortran.</td>
<td>For details, see 1.15.4.4 Installing NetCDF-Fortran.</td>
</tr>
</tbody>
</table>
1.15.4.1 Installing HDF5

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to load the compiler and Open MPI:

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH

export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
SLD_LIBRARY_PATH
```

**Step 3** Run the following command to decompress the HDF5 installation package:

```
tar zxvf hdf5-1.10.5.tar.gz
```

**Step 4** Run the following command to switch to the directory generated after decompression:

```
cd hdf5-1.10.1
```

**Step 5** Run the following commands to perform configuration:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
```

```
./configure --prefix=$HDF5 --enable-fortran --enable-static=yes --enable-parallel --enable-shared CC=$MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC
```

**Step 6** Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

**Step 7** Run the following commands to load environment variables:

```
export PATH=/path/to/HDF5/bin:$PATH

export LD_LIBRARY_PATH=/path/to/HDF5/lib:SLD_LIBRARY_PATH
```

----End

1.15.4.2 Installing PnetCDF

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to load the compiler and Open MPI. If the commands have been run, skip this step.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH

export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
SLD_LIBRARY_PATH
```

----End
Step 3 Run the following command to decompress the PnetCDF installation package:

```
tar zxvf pnetcdf-1.11.2.tar.gz
```

Step 4 Run the following command to switch to the directory generated after decompression:

```
 cd pnetcdf-1.11.2
```

Step 5 Run the following commands to perform configuration:

```
export MPICC=mpicc
export MPICXX=mpicxx
export MPIFC=mpifort
./configure --prefix=$PNETCDF
CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC" CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

Step 7 Run the following commands to load environment variables:

```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

---End

### 1.15.4.3 Installing NetCDF-C

**Procedure**

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install the dependency:

```
yum install curl libcurl libcurl-devel -y
```

Step 3 Run the following commands to load the compiler and Open MPI. If the commands have been run, skip this step.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:SLD_LIBRARY_PATH
```

Step 4 Decompress the NetCDF-C installation package.

```
tar zxvf netcdf-c-4.7.0.tar.gz
```

```
yum install curl.aarch64 libcurl.aarch64 libcurl-devel.aarch64 -y
```

Step 5 Run the following command to switch to the directory generated after decompression:
cd netcdf-c-4.7.0

**Step 6** Run the following commands to perform configuration:

- export MPICC=mpicc
- export MPICXX=mpicxx
- export MPIFC=mpifort

```
./configure --prefix=$NETCDF --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile CPPFLAGS=" -I${HDF5}/include -I${PNETCDF}/include" LDFLAGS=" -L $HDF5/lib -L$PNETCDF/lib -Wl,-rpath=$HDF5/lib -Wl,-rpath=$PNETCDF/lib " CFLAGS=" -L${HDF5}/lib -L${PNETCDF}/lib -I${HDF5}/include -I${PNETCDF}/include " CC=\$MPICC CXX=\$MPICXX FC=\$MPIFC F77=\$MPIFC
```

**Step 7** Run the following commands to perform compilation and installation:

- make -j 16
- make install

**Step 8** Run the following commands to load environment variables:

- export PATH=/path/to/NETCDF/bin:$PATH
- export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH

--- End

1.15.4.4 Installing NetCDF-Fortran

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to load the compiler and Open MPI. If the commands have been run, skip this step.

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
SLD_LIBRARY_PATH
```

**Step 3** Decompress the NetCDF-Fortran installation package.

```
tar zxfv netcdf-fortran-4.4.5.tar.gz
```

**Step 4** Run the following command to switch to the directory generated after decompression:

```
cd netcdf-fortran-4.4.5
```

**Step 5** Run the following commands to perform configuration:

- export MPICC=mpicc
- export MPICXX=mpicxx
- export MPIFC=mpifort
.configure --prefix=$NETCDF --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CPPFLAGS=" -I${HDF5}/include -I${NETCDF}/include" LDFLAGS=" -L${HDF5}/lib -L${NETCDF}/lib -Wl,-rpath=${HDF5}/lib -Wl,-rpath=${NETCDF}/lib" CFLAGS=" -I${HDF5}/include -I${NETCDF}/include" CC= $MPICC CXX=$MPICXX FC=$MPIFC F77=$MPIFC

Step 6 Run the following commands to perform compilation and installation:

make -j 16
make install

Step 7 Run the following commands to load environment variables:

export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH

1.15.5 Obtaining the Source Code

Procedure

Step 1 Download the SWAN installation package swan4131.tar.gz.
URL: http://swanmodel.sourceforge.net/download/zip/refrac.tar.gz

Step 2 Use SFTP to upload the SWAN installation package to the /path/to/SWAN directory on the server.

1.15.6 Compiling and Installing SWAN

Procedure

Step 1 Run the following commands to go to the installation directory and decompress the software package:

cd /path/to/SWAN
tar -zxvf swan4131.tar.gz
cd swan4131

Step 2 Run the following commands to modify the platform.pl file:

1. Open the platform.pl file.

   vim platform.pl

2. Press i to enter the editing mode and change the content of line 788 to

   $compiler = "gfortran".

   782 sub getcmpl { 783   my $compiler = $ENV('FC');
785 unless ($compiler) {
786   foreach (‘ifort’,’gfortran’,’f90’,’ifc’,’eifc’,’pgf90’,’xlf90’, ’lf95’,’g95’) {
787      $compiler = ”gfortran”;
788      my $path  = `which $compiler`;
789      last if $path;
790   }
791 }
792 }
793
794 return $compiler;
795 }

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 3 Run the following command to generate the configuration file:

    make config

Step 4 Run the following command to load the environment variable:

    export NETCDFROOT=/path/to/NETCDF

Step 5 Run the following command to generate the swan.exe file:

    make mpi

    ----End

1.15.7 Running and Verifying SWAN

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to create a test case directory and upload the test case package to the directory:

    mkdir -p /path/to/SWAN/swan-testcse
    cd /path/to/SWAN/swan-testcse
    tar -zxvf refrac.tar.gz

Step 3 Run the following commands to go to the directory and create a soft link:

    cd refract
    ln -s /path/to/SWAN/swan4131/swan.exe ./

Step 4 Run the following commands to run the case:

    cp a11refr.swn INPUT
    time mpirun --allow-run-as-root -np 40 ./swan.exe

Figure 1-19 shows an example of the CPU running time.
1.15.8 More Information

Wikipedia:

https://www.myroms.org/wiki/Getting_Started

1.16 CDO 1.9.8 Porting Guide (CentOS 7.6)

1.16.1 Introduction

The Climate Data Operators (CDO) software is a collection of many operators for standard processing of climate and numerical weather prediction (NWP) model output. The operators include simple statistical and arithmetic functions, data selection and subsampling tools, and spatial interpolation. CDO was developed to have the same set of processing functions for GRIB and netCDF datasets in one package.

Programming language: C++

Brief description: meteorological data processing

Open-source license: GPL
Recommended Version

CDO 1.9.8.

1.16.2 Environment Requirements

Hardware Requirements

Table 1-78 lists the hardware requirements.

Table 1-78 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 2-2 lists the software requirements.

Table 1-79 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDO</td>
<td>1.9.8</td>
<td><a href="https://code.mpimet.mpg.de/attachments/download/20826/cdo-1.9.8.tar.gz">https://code.mpimet.mpg.de/attachments/download/20826/cdo-1.9.8.tar.gz</a></td>
</tr>
<tr>
<td>GNU</td>
<td>9.3.0</td>
<td><a href="https://ftp.gnu.org/gnu/gcc/gcc-9.3.0/">https://ftp.gnu.org/gnu/gcc/gcc-9.3.0/</a></td>
</tr>
<tr>
<td>OPENMPI</td>
<td>4.0.3</td>
<td><a href="https://www.open-mpi.org/software/ompi/v4.0/">https://www.open-mpi.org/software/ompi/v4.0/</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.8.12-11</td>
<td>Provided by Huawei Yum</td>
</tr>
<tr>
<td>GRAB_API</td>
<td>1.12.3-5</td>
<td>Provided by the Huawei Yum source</td>
</tr>
<tr>
<td>NETCDF</td>
<td>4.3.3.1-5</td>
<td>Provided by the Huawei Yum source</td>
</tr>
</tbody>
</table>

OS Requirements

Table 1-80 lists the OS requirements.

Table 1-80 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
1.16.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CDO software porting.

Table 1-81 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/CDO</td>
<td>Installation path of CDO.</td>
<td>This installation path is only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

1.16.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-82 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>Configuring the Huawei Yum Source</td>
<td>For details, see <a href="#">1.16.4.2 Configuring the Huawei Yum Source</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Installing the Dependency Package</td>
<td>For details, see <a href="#">1.16.4.3 Installing the Dependency Package</a>.</td>
</tr>
</tbody>
</table>
1.16.4.1 Setting Up the Basic Environment

For details, see "Setting Up the Environment for the Cluster Scenario" in HPC Solution Basic Environment Setup Guide.

1.16.4.2 Configuring the Huawei Yum Source

1.16.4.2.1 Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to back up the Base source file:

```
cp -a /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.bak
```

**Step 3** Run the following command to download the new CentOS-Base.repo file to the /etc/yum.repos.d/ directory:

```
wget -O /etc/yum.repos.d/CentOS-Base.repo https://mirrors.huaweicloud.com/repository/conf/CentOS-AltArch-7.repo
```

**Step 4** Run the following command to clear the original Yum cache:

```
yum clean all
```

**Step 5** Run the following commands to refresh the Yum source cache:

```
yum makecache
yum repolist all
```

**Step 6** Run the following command to back up the EPEL source file:

```
cp -a /etc/yum.repos.d/epel.repo /etc/yum.repos.d/epel.repo.backup
```

**Step 7** Run the following command to modify the epel.repo file:

```
sed -i "s/#baseurl/baseurl/g" /etc/yum.repos.d/epel.repo
sed -i "s/#metalink/metalink/g" /etc/yum.repos.d/epel.repo
sed -i "s@https://download.fedoraproject.org/pub@https://mirrors.huaweicloud.com@g" /etc/yum.repos.d/epel.repo
```

**Step 8** Run the following command to update the Yum source:

```
yum update
```

----End

1.16.4.3 Installing the Dependency Package

1.16.4.3.1 Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to install all the dependencies:
yum install nco netcdf netcdf-devel hdf hdf-devel hdf5 hdf5-devel grib_api grib_api-devel -y
----End

1.16.5 Obtaining Source Code

Procedure

Step 1  Download the CDO installation package `cdo-1.9.8.tar.gz` from the following link:
        https://code.mpimet.mpg.de/projects/cdo/

Step 2  Use an SFTP tool to upload the CDO installation package to the `/path/to/CDO` directory on the server.
        ----End

1.16.6 Compilation and Installation

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to go to the main program installation directory:
        `cd /path/to/CDO`

Step 3  Run the following command to decompress the package:
        `tar -zxvf cdo-1.9.8.tar.gz`

Step 4  Run the following command to go to the directory generated after decompression:
        `cd cdo-1.9.8/`

Step 5  Run the following command to perform configuration:
        `./configure --prefix=/path/to/CDO --with-netcdf=/usr --with-hdf5=/usr --with-grib_api=/usr`

Step 6  Run the following commands to perform compilation:
        `make -j 16`
        `make install`
        ----End

1.16.7 Running and Verifying CDO

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Upload the outputted file `wrfout_d01_2005-06-04_09_00_00` to the test directory.
NOTE

wrfout_d01_2005-06-04_09_00_00 is the outputted result file of executing the WRF count2.5 test case. For details, see WRF 3.8.1 Porting Guide (CentOS 7.6).

Step 3  Run the following command to perform verification:

```
export PATH=/path/to/CDO/bin:$PATH
cdo info wrfout_d01_2005-06-04_09_00_00
```

Figure 1-20 An example output

```
011 : 2005-06-04 09:00:00   35 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
011 : 2005-06-04 09:00:00   36 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
012 : 2005-06-04 09:00:00   31 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
012 : 2005-06-04 09:00:00   32 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
013 : 2005-06-04 09:00:00   34 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
013 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
013 : 2005-06-04 09:00:00   31 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
014 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
015 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
016 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
017 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
018 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
019 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
020 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
021 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
022 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
023 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
024 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
025 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
026 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
027 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
100 : 2005-06-04 09:00:00   30 1800000  0 : 6.0000 0.00000 4.00000 1.0000 -100
```

----End

1.16.8 More Resources

To obtain more resources, visit the CDO official website:

https://code.mpimet.mpg.de/projects/cdo

1.17 ecCodes 2.18.0 Porting Guide (CentOS 7.6)

1.17.1 Introduction

ecCodes is a package developed by ECMWF which provides an application programming interface and a set of tools for decoding and encoding messages in the following formats:

- WMO FM-92 GRIB edition 1 and edition 2
- WMO FM-94 BUFR edition 3 and edition 4
- WMO GTS abbreviated header (only decoding)
A useful set of command line tools provide quick access to the messages. C, Fortran 90 and Python interfaces provide access to the main ecCodes functionality.

ecCodes is an evolution of GRIB-API. It is designed to provide the user with a simple set of functions to access data from several formats with a key/value approach.

Programming languages: C/Fortran

Brief description: Provides an application programming interface and a set of tools for decoding and encoding messages in WMO GRIB and BUFR formats.

Open-source license: Apache License 2.0

**Recommended Version**

ecCodes 2.18.0.

### 1.17.2 Environment Requirements

#### Hardware Requirements

Table 1-83 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

#### Software Requirements

Table 2-2 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecCodes</td>
<td>2.18.0</td>
<td><a href="https://confluence.ecmwf.int/download/attachments/45757960/eccodes-2.18.0-Source.tar.gz">https://confluence.ecmwf.int/download/attachments/45757960/eccodes-2.18.0-Source.tar.gz</a></td>
</tr>
<tr>
<td>CMAKE</td>
<td>3.16.4</td>
<td><a href="https://cmake.org/files/v3.16/cmake-3.16.4.tar.gz">https://cmake.org/files/v3.16/cmake-3.16.4.tar.gz</a></td>
</tr>
</tbody>
</table>

#### OS Requirements

Table 1-85 lists the OS requirements.
### 1.17.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ecCodes software porting.

#### Table 1-86 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in <strong>HPC Solution Basic Environment Setup Guide</strong>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/ECCODES</td>
<td>Installation path of ecCodes.</td>
<td>The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake.</td>
<td></td>
</tr>
</tbody>
</table>

### 1.17.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

#### Configuration Process

#### Table 1-87 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <strong>HPC Solution Basic Environment Setup Guide</strong>.</td>
</tr>
</tbody>
</table>
1.17.4.1 Installing CMake

1.17.4.1.1 Procedure

- **Step 1** Use PuTTY to log in to the server as the root user.
- **Step 2** Run the following command to decompress the package:
  
  ```
tar -xvf cmake-3.16.4.tar.gz
  ```
- **Step 3** Run the following command to switch to the directory generated after decompression:
  
  ```
  cd cmake-3.16.4
  ```
- **Step 4** Run the following commands to perform compilation and installation:
  
  ```
  ./bootstrap --prefix=/path/to/CMAKE
  make -j 16
  make install
  ```
- **Step 5** Run the following command to set environment variables:
  
  ```
  export PATH=/path/to/CMAKE/bin:$PATH
  ```

----End

1.17.5 Obtaining Source Code

Procedure

- **Step 1** Download the ecCodes installation package `eccodes-2.18.0-Source.tar.gz` from the following link:
  
  ```
  https://confluence.ecmwf.int/download/attachments/45757960/eccodes-2.18.0-Source.tar.gz
  ```
- **Step 2** Use an SFTP tool to upload the ecCodes installation package to the `/path/to/ECCODES` directory on the server.

----End

1.17.6 Compilation and Installation

Procedure

- **Step 1** Use PuTTY to log in to the server as the root user.
- **Step 2** Run the following command to install dependencies:
  
  ```
  yum install openjpeg-devel jasper-devel python-devel numpy -y
  ```
- **Step 3** Run the following command to go to the main program installation directory:
cd /path/to/ECCODES

**Step 4** Run the following command to decompress the package:

```bash
tar -xvf eccodes-2.18.0-Source.tar.gz
```

**Step 5** Run the following command to switch to the directory generated after decompression:

```bash
cd eccodes-2.18.0-Source
```

**Step 6** Run the following commands to create the build directory.

```bash
mkdir build
cd build
```

**Step 7** Run the following command:

```bash
mkdir build

cd build

``` cmake ..../eccodes-2.18.0-Source -DCMAKE_INSTALL_PREFIX=/path/to/ECCODES
```

**Step 8** Run the following commands to perform compilation, testing, and installation:

```bash
make -j 16
ctest
make install
```

**Step 9** Run the following command to set environment variables:

```bash
export PATH=/path/to/ECCODES/bin:$PATH
```

```bash
-----End
```

### 1.17.7 Running and Verifying ecCodes

**Procedure**

Run the following command to perform running and verification:

```bash
code_info
```

If the following information is displayed, the installation is successful:

```
[root@kunpengnode21 eccodes]# code_info
eccodes Version 2.18.0
```

### 1.17.8 More Resources

ecCodes official website:

https://software.ecmwf.int/wiki/display/ECC/ecCodes+Home
### 1.18 ecFlow 5.5.2 Porting Guide (CentOS 7.6)

#### 1.18.1 Introduction

ecFlow is a client/server workflow package that enables users to run a large number of programs (with dependencies on each other and on time) in a controlled environment. It provides reasonable tolerance for hardware and software failures, combined with restart capabilities.

ecFlow runs as a server receiving requests from clients. The command-line interface (CLI), the graphical interface, scripts and the Python API (application interface) are the clients. The server is based on C++/boost ASIO and uses TCP/IP for communication. Multiple servers can be run on the same hardware. ecFlow submits tasks (jobs) and receives acknowledgements from tasks via specific commands embedded in the scripts. The relationship between tasks is stored in ecFlow, and it is able to submit tasks dependent on the status of other tasks and attributes like time.

The CLI for ecFlow allows the suite definition to be loaded and retrieved from the server. Also it provides a rich set of commands for communication with the server.

The Python API allows the entire suite definition structure to be specified and loaded into the server. A suite is a collection of interrelated tasks. In ecFlow suites are described by a definition file. The Python API also provides functionality for client to server communication. In addition, it allows checking of the suite, testing the defined interrelations between tasks, and other references and limits.

Programming language: C++

Brief description: A client/server workflow package that enables users to run a large number of programs (with dependencies on each other and on time) in a controlled environment.

Open-source license: Apache License 2.0

**Recommended Version**

ecFlow 5.5.2.

#### 1.18.2 Environment Requirements

##### Hardware Requirements

Table 1-88 lists the hardware requirements.

**Table 1-88** Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
Software Requirements

Table 2-2 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecFlow</td>
<td>5.5.2</td>
<td><a href="https://confluence.ecmwf.int/download/attachments/8650755/ecFlow-5.5.2-Source.tar.gz">https://confluence.ecmwf.int/download/attachments/8650755/ecFlow-5.5.2-Source.tar.gz</a></td>
</tr>
<tr>
<td>boost</td>
<td>1.72.0</td>
<td><a href="https://dl.bintray.com/boostorg/release/1.72.0/source/boost_1_72_0.tar.gz">https://dl.bintray.com/boostorg/release/1.72.0/source/boost_1_72_0.tar.gz</a></td>
</tr>
<tr>
<td>CMAKE</td>
<td>3.16.4</td>
<td><a href="https://cmake.org/files/v3.16/cmake-3.16.4.tar.gz">https://cmake.org/files/v3.16/cmake-3.16.4.tar.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 1-90 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

1.18.3 Paths for Software Porting

This section describes the software installation paths involved in the ecFlow software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/ECFLOW_BUILD</td>
<td>Installation path of ecFlow and Boost.</td>
<td>The installation paths are only examples. Shared paths are recommended. Replace the paths</td>
</tr>
</tbody>
</table>
1.18.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 1-92 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

1.18.4.1 Installing CMake

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the package:

```
tar -xvf cmake-3.16.4.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:

```
cd cmake-3.16.4
```

**Step 4** Run the following command to perform compilation and installation:

```
./bootstrap --prefix=/path/to/CMAKE
make -j 16
make install
```

**Step 5** Run the following command to set environment variables:
1.18.5 Obtaining Source Code

**Procedure**

**Step 1** Download the ecFlow installation package `ecFlow-5.5.2-Source.tar.gz` and the Boost installation package `boost_1_72_0.tar.gz`.

ecFlow is available at: [https://confluence.ecmwf.int/download/attachments/8650755/ecFlow-5.5.2-Source.tar.gz](https://confluence.ecmwf.int/download/attachments/8650755/ecFlow-5.5.2-Source.tar.gz)

Boost is available at: [https://dl.bintray.com/boostorg/release/1.72.0/source/boost_1_72_0.tar.gz](https://dl.bintray.com/boostorg/release/1.72.0/source/boost_1_72_0.tar.gz)

**Step 2** Use an SFTP tool to upload the ecFlow and Boost installation packages to the `/path/to/ECFLOW_BUILD` directory on the server.

1.18.6 Compilation and Installation

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to go to the main program installation directory:

```
cd /path/to/ECFLOW_BUILD
```

**Step 3** Run the following commands to decompress the Boost installation package:

```
tar -xvf boost_1_72_0.tar.gz
```

**Step 4** Run the following command to decompress the ecFlow installation package:

```
tar -xvf ecFlow-5.5.2-Source.tar.gz
```

**Step 5** Run the following commands to create the two environment variables:

```
export WK=/path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source
export BOOST_ROOT=/path/to/ECFLOW_BUILD/boost_1_72_0
```

**Step 6** Run the following commands to go to the Boost installation directory and build the Boost library:

```
cd $BOOST_ROOT
./bootstrap.sh
./b2 install --prefix=/path/to/ECFLOW_BUILD/boost_1_72_0
```

**Step 7** Run the following command to compile the Boost library using ecFlow:

```
$WK/build_scripts/boost_build.sh
```
Step 8  Run the following command to go to the directory generated after decompression:

```
cd $WK
```

Step 9  Run the following commands to create an independent directory for ecFlow:

```
mkdir build
cd build
```

Step 10  Run the following commands to perform compilation and installation:

```
cmake .. -DENABLE_UI=OFF -DENABLE_PYTHON=OFF
make -j 16
make install
```

----End

1.18.7 Running and Verifying ecFlow

Procedure

Step 1  Run the following command to go to the test directory:

```
cd /path/to/ECFLOW_BUILD/ecFlow-5.5.2-Source/build
```

Step 2  Run the following command to perform the test:

```
make test
```

Figure 1-21 shows an example of the result.

Figure 1-21 An example result

```
Running tests...
Test project /path/to/ecflow_build/ecFlow-5.5.2-Source/build
 Start 1: u_a/core
 1/8 Test #1: u_a/core --------------------------------- Passed  0.22 sec
 Start 2: u_anattr
 2/8 Test #2: u_anattr --------------------------------- Passed  0.20 sec
 Start 3: u_anode
 3/8 Test #3: u_anode --------------------------------- Passed  0.16 sec
 Start 4: u_aparser
 4/8 Test #4: u_aparser --------------------------------- Passed  0.13 sec
 Start 5: u_base
 Start 6: c_csim
 6/8 Test #5: c_csim --------------------------------- Passed  0.80 sec
 Start 7: s_client
 7/8 Test #7: s_client --------------------------------- Passed  0.89 sec
 Start 8: u_server
 8/8 Test #8: u_server --------------------------------- Passed  0.06 sec
100% tests passed, 0 tests failed out of 8
Label Time Summary:
eflow = 98.48 sec*proc (8 tests)
exeutable = 98.48 sec*proc (8 tests)
Total Test time (real) = 98.50 sec
```

----End
1.18.8 More Resources

ecFlow official website:

https://confluence.ecmwf.int/display/ECFLOW/ecflow+home

1.19 NCEPLIBS 1.2.0 Porting Guide (CentOS 7.6)

1.19.1 Introduction

NCEPLIBS provides the dependency libraries for the meteorological field. It is widely used to build dependency relationships for meteorological software such as weather models or weather applications.

For more information about NCEPLIBS, visit the NCEPLIBS official website.

Programming languages: C/Fortran

Brief description: common dependency libraries in the meteorological field

Open-source license: Apache

Recommended Version

NCEPLIBS v1.2.0.

1.19.2 Environment Requirements

Hardware Requirements

Table 1-93 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 1-94 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCEPLIBS</td>
<td>v1.3.0</td>
<td><a href="https://github.com/NOAA-EMC/NCEPLIBS">https://github.com/NOAA-EMC/NCEPLIBS</a></td>
</tr>
<tr>
<td>GNU</td>
<td>9.1.0</td>
<td><a href="https://ftp.gnu.org/gnu/gcc/gcc-9.1.0/">https://ftp.gnu.org/gnu/gcc/gcc-9.1.0/</a></td>
</tr>
<tr>
<td>Item</td>
<td>Version</td>
<td>Download Link</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OPENMPI</td>
<td>4.0.1</td>
<td><a href="https://www.open-mpi.org/software/ompi/v4.0/">https://www.open-mpi.org/software/ompi/v4.0/</a></td>
</tr>
<tr>
<td>cmake</td>
<td>3.17.3</td>
<td><a href="https://github.com/Kitware/CMake/releases/download/v3.17.3/cmake-3.17.3.tar.gz">https://github.com/Kitware/CMake/releases/download/v3.17.3/cmake-3.17.3.tar.gz</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.1</td>
<td><a href="https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/">https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/</a></td>
</tr>
<tr>
<td>NETCDF</td>
<td>4.4.1.1</td>
<td><a href="https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1">https://github.com/Unidata/netcdf-c/releases/tag/v4.4.1.1</a></td>
</tr>
<tr>
<td>NETCDF-F</td>
<td>4.4.1</td>
<td><a href="https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1">https://github.com/Unidata/netcdf-fortran/releases/tag/v4.4.1</a></td>
</tr>
</tbody>
</table>

**OS Requirements**

Table 1-95 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

**1.19.3 Paths for Software Porting**

This chapter lists the software installation paths involved in the NCEPLIBS software porting.

<table>
<thead>
<tr>
<th>N o.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>
### 1.19.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

#### Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Installing HDF5</td>
<td>For details, see <a href="#">1.19.4.1 Installing HDF5</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Installing PNETCDF</td>
<td>For details, see <a href="#">1.19.4.2 Installing PNETCDF</a>.</td>
</tr>
<tr>
<td>4</td>
<td>Installing NETCDF</td>
<td>For details, see <a href="#">1.19.4.3 Installing NETCDF</a>.</td>
</tr>
<tr>
<td>5</td>
<td>Installing NETCDF-FORTRAN</td>
<td>For details, see <a href="#">1.19.4.4 Installing NETCDF-FORTRAN</a>.</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>/path/to/NCEPLIBS</td>
<td>Installation path of NCEPLIBS.</td>
<td>The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/PNETCDF</td>
<td>Installation path of PNETCDF 1.11.2.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NETCDF 4.4.1.1.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installation path of NETCDF-Fortran 4.4.1.</td>
<td></td>
</tr>
</tbody>
</table>
1.19.4.1 Installing HDF5

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to install HDF5.

```bash
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
SLD_LIBRARY_PATH

tar -xvf hdf5-1.10.1.tar.gz

cd hdf5-1.10.1

mkdir -p /path/to/HDF5

./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort

make -j 16

make install

----End
```

1.19.4.2 Installing PNETCDF

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to install PNETCDF.

```bash
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
SLD_LIBRARY_PATH

tar -xvf parallel-netcdf-1.11.2.tar.gz

cd parallel-netcdf-1.11.2

mkdir -p /path/to/PNETCDF

./configure --prefix=/path/to/PNETCDF --build=aarch64-unknown-linux-gnu
CFLAGS="-fPIC -DPIC" CXXFLAGS="-fPIC -DPIC" FCFLAGS="-fPIC" FFLAGS="-fPIC" CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort

make -j 16

make install

----End
```
1.19.4.3 Installing NETCDF

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to install NETCDF:

```bash
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:SPATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
SLD_LIBRARY_PATH
yum install curl* -y
tar -xvf netcdf-4.4.1.1.tar.gz
cd netcdf-4.4.1.1
mkdir -p /path/to/NETCDF
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-pnetcdf --enable-largefile CC=mpicc CXX=mpicxx
FC=mpifort F77=mpifort CPPFLAGS="-I/path/to/HDF5/include -l|path/to/PNETCDF|include" LDFLAGS="-L/path/to/HDF5|lib -L/path/to/PNETCDF|lib -Wl,-rpath=/path/to/HDF5|lib -Wl,-rpath=/path/to/PNETCDF|lib" CFLAGS="-L/path/to/HDF5|lib -L/path/to/PNETCDF|lib -L/path/to/HDF5/include -I/path/to/PNETCDF|include"
make -j 16
make install
```

----End

1.19.4.4 Installing NETCDF-FORTRAN

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to install NETCDF-FORTRAN:

```bash
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:SPATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
SLD_LIBRARY_PATH
tar -xvf netcdf-fortran-4.4.1.tar.gz
cd netcdf-fortran-4.4.1
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --with-pic --disable-doxygen --enable-largefile --enable-static CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort CPPFLAGS="-I/path/to/HDF5/include -l|path/to/PNETCDF|include" LDFLAGS="-L/path/to/HDF5|lib -L/path/to/PNETCDF|lib -Wl,-rpath=/path/to/HDF5|lib -Wl,-rpath=/path/to/PNETCDF|lib" CFLAGS="-L/path/to/HDF5|lib -L/path/to/PNETCDF|lib -L/path/to/HDF5/include -I/path/to/PNETCDF|include"
make -j 16
make install
```

----End
make -j 16
make install

----End

1.19.4.5 Installing CMake

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following command to decompress the CMake installation package:
   `tar zxvf cmake-3.17.3.tar.gz`
Step 3 Run the following command to go to the directory generated after decompression:
   `cd cmake-3.17.3`
Step 4 Run the following commands to perform compilation and installation:
   `./configure -prefix=/path/to/CMAKE`
   `make -j 32`
   `make install`
Step 5 Run the following command to set the environment variables of CMake:
Step 6 `export PATH=/path/to/CMAKE/bin:$PATH`

----End

1.19.5 Obtaining Source Code

Procedure

Step 1 Obtain the NCEPLIBS package NCEPLIBS-1.2.0.tar.gz from the following link:
   [https://github.com/NOAA-EMC/NCEPLIBS](https://github.com/NOAA-EMC/NCEPLIBS)
Step 2 Use an SFTP tool to upload the downloaded installation package to the /path/to/NCEPLIBS directory on the server.

----End

1.19.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following command to go to the main program installation directory:
   `cd /path/to/NCEPLIBS`
Step 3 Run the following command to create the build and nceplibs directories, and go to the build directory:
   `mkdir -p build && mkdir nceplibs && cd build`
Step 4 Run the following commands to build the software:
```
cmake ../ -DCMAKE_INSTALL_PREFIX=../nceplibs
```

Step 5 Run the following command to perform installation:
```
make -j 16
```

**NOTE**

If an error is reported during the installation, see 1.19.8 Troubleshooting for the solution.

----End

### 1.19.7 Running and Verifying NCEPLIBS

**Procedure**

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the library file installation directory:
```
cd /path/to/NCEPLIBS/NCEPLIBS
```

Step 3 Run the following command to view the library file:
```
ll
```

If the following dependency library information is displayed, the installation is complete:

![Figure 1-22 Dependency libraries](image)

----End
1.19.8 Troubleshooting

Problem 1: An Error Is Reported During Software Build

Symptom:

If the following error is reported during the installation, add the NetCDF installation path to the specified file based on the error information.

```
Found sig: /usr/x86_64-linux-gnu/lib/libsigsegv.so.1
Found sig: /usr/x86_64-linux-gnu/lib/libsigsegv.so.1
Found sig: /usr/x86_64-linux-gnu/lib/libsigsegv.so.1
Found sig: /usr/x86_64-linux-gnu/lib/libsigsegv.so.1
Found sig: /usr/x86_64-linux-gnu/lib/libsigsegv.so.1
Could not find NetCDF (missing: NetCDF_INCLUDE_DIR)
```

Possible Cause:

The NetCDF installation path is not added to the specified file.

Procedure:

**Step 1** Run the following commands to modify the `CmakCache.txt` file:

1. Run the following command to modify the `CmakCache.txt` file:
   ```
   vi /path/to/NCEPLIBS/NCEPLIBS1.3.0/build/w3emc/src/w3emc-build/CmakCache.txt
   ```

2. Press `i` to go to the edit mode.

   Before the modification:
   ```
   //NetCDF C include directory
   NetCDF_C_INCLUDE_FILE:FILEPATH=NetCDF_C_INCLUDE_FILE-NOTFOUND
   ```

   After the modification:
   ```
   //NetCDF C include directory
   NetCDF_C_INCLUDE_FILE:FILEPATH=/path/to/NETCDF/include
   ```

3. Press `Esc`, type `:wq!`, and press `Enter` to save the file and exit.

**Step 2** Run the following command to build the software. If a similar error occurs, perform the previous steps to rectify the error.

```
make -j 16
```

----End

1.19.9 More Resources

For details about how to install NCEPLIBS, visit the following website:

```
https://github.com/NOAA-EMC/NCEPLIBS%0c
```

1.20 PyFerret 7.6.0 Porting Guide (CentOS 7.6)
1.20.1 Introduction

Ferret is an interactive computer visualization and analysis environment designed to meet the needs of oceanographers and meteorologists analyzing large and complex gridded data sets.

Programming languages: C/Fortran/Python

Brief description: ecosystem dynamics modeling software

Open-source license: GPL

1.20.2 Environment Requirements

Hardware Requirements

Table 1-98 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 1-99 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyFerret</td>
<td>7.6.0</td>
<td><a href="https://github.com/NOAA-PMEL/PyFerret/releases/tag/v7.6.0">https://github.com/NOAA-PMEL/PyFerret/releases/tag/v7.6.0</a></td>
</tr>
<tr>
<td>GNU</td>
<td>9.3.0</td>
<td><a href="https://ftp.gnu.org/gnu/gcc/gcc-9.1.0/">https://ftp.gnu.org/gnu/gcc/gcc-9.1.0/</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.6</td>
<td><a href="https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.6/src/">https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.6/src/</a></td>
</tr>
<tr>
<td>NETCDF</td>
<td>4.7.3</td>
<td><a href="https://github.com/Unidata/netcdf-c/releases/tag/v4.7.3">https://github.com/Unidata/netcdf-c/releases/tag/v4.7.3</a></td>
</tr>
<tr>
<td>NETCDF-F</td>
<td>4.5.2</td>
<td><a href="https://github.com/Unidata/netcdf-fortran/releases/tag/v4.5.2">https://github.com/Unidata/netcdf-fortran/releases/tag/v4.5.2</a></td>
</tr>
</tbody>
</table>
OS Requirements

Table 1-100 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

1.20.3 Paths for Software Porting

This chapter lists the software installation paths involved in the PyFerret software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/PyFerret</td>
<td>Installation path of PyFerret.</td>
<td>The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NETCDF 4.7.3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installation path of NETCDF-Fortran 4.5.2.</td>
<td></td>
</tr>
</tbody>
</table>

1.20.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.
**Configuration Process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>Configure the Yum source and install the dependencies.</td>
<td>For details, see <a href="#">1.20.4.2 Configuring the Yum Source and Installing the Dependencies</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Installing HDF5</td>
<td>For details, see <a href="#">1.20.4.3 Installing HDF5</a>.</td>
</tr>
<tr>
<td>4</td>
<td>Installing NETCDF</td>
<td>For details, see <a href="#">1.20.4.4 Installing NETCDF</a>.</td>
</tr>
<tr>
<td>5</td>
<td>Installing NETCDF-FORTRAN</td>
<td>For details, see <a href="#">1.20.4.5 Installing NETCDF-FORTRAN</a>.</td>
</tr>
</tbody>
</table>

### 1.20.4.1 Setting Up the Basic Environment

Install the GNU. For details, see "Setting Up the Environment for the Cluster Scenario" in [HPC Solution Basic Environment Setup Guide](#).

### 1.20.4.2 Configuring the Yum Source and Installing the Dependencies

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following commands to configure the basic Yum source:

```bash
cd /etc/yum.repos.d/
mv CentOS-Base.repo CentOS-Base.repo-bak
wget -O /etc/yum.repos.d/CentOS-Base.repo https://mirrors.huaweicloud.com/repository/conf/CentOS-AltArch-7.repo
```

**Step 3** Run the following command to configure the additional Yum source:

```bash
vim epel.repo
```

Add the following content to the file:

```bash
[epel]
name=Extra Packages for Enterprise Linux 7 - $basearch
baseurl=http://mirrors.tools.huawei.com/epel/7/$basearch
failovermethod=priority
enabled=1
gpgcheck=0
```
Step 4  Run the following commands to clear the Yum cache:
   yum clean all
   yum makecache

Step 5  Run the following command to install the dependencies:
   yum install python numpy libgfortran python-qt4 scipy pyshp -y

---End

1.20.4.3 Installing HDF5

Procedure

Step 1  Use PuTTY to log in to the server as the root user.
Step 2  Run the following commands to install HDF5.
   export PATH=/path/to/GNU/bin:SPATH
   export LD_LIBRARY_PATH=/path/to/GNU/lib64:SLD_LIBRARY_PATH
   tar -xvf hdf5-1.10.6.tar.gz
   cd hdf5-1.10.6
   mkdir -p /path/to/HDF5
   ./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-shared CC=gcc CXX=g++
   FC=gfortran F77=gfortran CFLAGS=-fPIC
   make -j 16
   make install

---End

1.20.4.4 Installing NETCDF

Procedure

Step 1  Use PuTTY to log in to the server as the root user.
Step 2  Run the following commands to install NETCDF.
   export PATH=/path/to/GNU/bin:SPATH
   export LD_LIBRARY_PATH=/path/to/GNU/lib64:SLD_LIBRARY_PATH
   yum install curl* -y
   tar -xvf netcdf-4.7.3.tar.gz
   cd netcdf-c-4.7.3
   mkdir -p /path/to/NETCDF
1.20.4.5 Installing NETCDF-FORTRAN

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install NETCDF-FORTRAN.

```bash
export PATH=/path/to/GNU/bin: $PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64: $LD_LIBRARY_PATH
tar -xvf netcdf-fortran-4.5.2.tar.gz
cd netcdf-fortran-4.5.2
./configure --prefix=/path/to/NETCDF --build=aarch64-unknown-linux-gnu --enable-shared --enable-netcdf-4 --enable-dap --with-pic --disable-doxygen --enable-static --enable-largefile CC=gcc CXX=g++ FC=gfortran F77=gfortran CPPFLAGS="-I/path/to/HDF5/include" LDFLAGS="-L/path/to/HDF5/lib -Wl,-rpath=/path/to/HDF5/lib " CFLAGS="-L/path/to/HDF5/lib -L/path/to/HDF5/ include"

make -j 16
make install
```

----End

1.20.5 Obtaining Source Code

Procedure

Step 1 Obtain the PyFerret installation package PyFerret-7.6.0.tar.gz from the following link:

https://github.com/NOAA-PMEL/PyFerret/releases/tag/v7.6.0
Step 2 Use an SFTP tool to upload PyFerret-7.6.0.tar.gz to the /path/to/PyFerret directory on the server.

--- End

1.20.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the main program installation directory:

```
cd /path/to/PyFerret
```

Step 3 Run the following commands to install the basic dependencies:

```
yum -y install cairo*
```

```
yum -y install pango-devel pangomm pango-tests pangomm-devel pangomm-doc
```

Step 4 Run the following command to decompress the package:

```
tar -zxvf PyFerret-7.6.0.tar.gz
```

Step 5 Run the following command to switch to the directory generated after decompression:

```
cd PyFerret-7.6.0
```

Step 6 Run the following command to copy the configuration file:

```
 cp site_specific.mk.in site_specific.mk
```

Step 7 Run the following command to modify the site_specific.mk file:

```
vim site_specific.mk
```

Add the following content to line 10:

```
HOME=/path/to/PyFerret
```

Modify the following information:

**Before the modification:**

```
DIR_PREFIX = $(HOME)/git/PyFerret
INSTALL_FER_DIR = $(HOME)/PyFerret
BUILDTYPE = x86_64-linux
PYTHON_EXE = python3.6
HDF5_LIBDIR = /usr/local/hdf5/lib
NETCDF_LIBDIR = /usr/local/netcdf/lib
```

**After the modification:**

```
DIR_PREFIX = $(HOME)/PyFerret-7.6.0
INSTALL_FER_DIR = $(HOME)/pyferret
BUILDTYPE = aarch64-linux
PYTHON_EXE = python2.7
HDF5_LIBDIR = /path/to/HDF5/lib
NETCDF_LIBDIR = /path/to/NETCDF/lib
```
Step 8 Run the following command to copy the configuration file:

```
cp platform_specific.mk.x86_64-linux platform_specific.mk.aarch64-linux
```

Step 9 Run the following command to modify `platform_specific.mk.aarch64-linux`:

```
sed -i "s/-m64//g" platform_specific.mk.aarch64-linux
```

Step 10 Run the following command to go to the `ef_utility` directory:

```
cd external_functions/ef_utility/
```

Step 11 Run the following command to copy the configuration file:

```
cp site_specific.mk.in site_specific.mk
```

Step 12 Run the following command to modify the `site_specific.mk` file:

```
vim site_specific.mk
```

Before the modification:

- BUILDTYPE = x86_64-linux
- PYTHON_EXE = python3.6

After the modification:

- BUILDTYPE = aarch64-linux
- PYTHON_EXE = python2.7

Step 13 Run the following command to copy the configuration file:

```
cp platform_specific.mk.x86_64-linux platform_specific.mk.aarch64-linux
```

Step 14 Run the following command to modify the `platform_specific.mk.aarch64-linux` file:

```
sed -i "s/-m64//g" platform_specific.mk.aarch64-linux
```

Step 15 Run the following command to go back to the original directory:

```
cd -
```

Step 16 Run the following commands to start the installation:

```
make
make install
```

1.20.7 Running and Verifying PyFerret

Procedure

Step 1 Obtain the PyFerret installation package `FerretDatasets-7.6.tar.gz` from the following link:

```
https://github.com/NOAA-PMEL/FerretDatasets/archive/v7.6.tar.gz
```

Step 2 Use an SFTP tool to upload `FerretDatasets-7.6.tar.gz` to the `/path/to/PyFerret` directory on the server.
Step 3 Use PuTTY to log in to the server as the root user.

Step 4 Run the following command to decompress FerretDatasets-7.6.tar.gz:

```
tar -zxvf FerretDatasets-7.6.tar.gz
```

Step 5 Run the following command to go to the PyFerret installation directory:

```
cd pyferret/bin
```

Step 6 Run the Finstall script to generate a file.

```
./Finstall
```

**NOTE**

During the process, you need to type the PyFerret installation path, FerretDatasets package path, and ferret_paths.sh script generation path. This process is an interactive process, and you need to type the needed information at the following prompts:

- **Prompt:** Proceed? (y/n): Type y and press Enter to continue.
- **FER_DIR:** PyFerret installation path. Type /path/to/PyFerret/.
- **FER_DSETS:** FerretDatasets package path. Type /path/to/PyFerret/FerretDatasets-7.6.
- **desired ferret_paths location:** ferret_paths.sh script path. Type /path/to/PyFerret/.
- **ferret_paths link to create?** (c/s/n): Type n.
- **python executable to use:** Type python.

Step 7 Run the following command to load the environment variables:

```
source ferret_paths.sh
```

Step 8 Run the following command to start the main program.

```
./pyferret
```

```
yes? go /help mp_demo
yes? USE coads_climatology
yes? SHADE/X=0:360/Y=-90:90/L=1/TITLE="Standard Plot" sst
yes? GO fland
yes? GO land
yes? exit
```
1.20.8 More Resources

To obtain more resources, visit the PyFerret official website at https://ferret.pmel.noaa.gov/Ferret/.

1.21 ncl 6.3.0 Porting Guide (CentOS 7.6)

1.21.1 Introduction

ncl is designed specifically for scientific data processing and visualization, widely used in the area of atmospheric modeling.

Programming languages: C/Fortran

Brief description: data processing and visualization

Recommended Version

ncl-6.3.0.
1.21.2 Environment Requirements

Hardware Requirements

Table 1-103 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

OS Requirements

Table 1-104 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
</tbody>
</table>

1.21.3 Paths for Software Porting

This chapter lists the software installation paths involved in the ncl software porting.

Table 1-105 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in <a href="https://www.centos.org/download/">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>Yum is installed in the system directory.</td>
<td>This installation path is only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>
1.21.4 Configuring the Compilation Environment

Procedure

Step 1 Configure the Yum network source. For details, see the Huawei mirror source at https://mirrors.huaweicloud.com/.

Step 2 Configure the CentOS-AltArch and EPEL mirrors.

Step 3 Run the following command to check whether the configuration is successful:

```
yum repolist
```

If information similar to the following information is displayed, the configuration is successful.

```
repo_id          repo name
base/aarch64    CentOS-7 - Base - mirrors.tools.huawei.com
epel/aarch64    Extra Packages for Enterprise Linux 7 - aarch64
extras/aarch64   CentOS-7 - Extras - mirrors.tools.huawei.com
updates/aarch64  CentOS-7 - Updates - mirrors.tools.huawei.com
repolist: 21,SQL
```

----End

1.21.5 Compilation and Installation

Procedure

--- CAUTION ---

Use the GCC embedded in the OS to perform the installation. Do not load the GCC 9.3 environment.

--- End ---

Step 1 Run the following command to install ncl and its dependencies:

```
yum install ncl.aarch64 ncl-common.noarch ncl-devel.aarch64 ncl-examples.noarch -y
```

Step 2 Run the following commands to set environment variables:

```
export NCARG_FONTCAPS=/usr/lib64/ncarg/fontcaps
export NCARG_GRAPHCAPS=/usr/lib64/ncarg/graphcaps
export NCARG_ROOT=/usr
export NCARG_DATABASE=/usr/lib64/ncarg/database
export NCARG_LIB=/usr/lib64/ncarg
export NCARG_NCARG=/usr/share/ncarg
export EXTRA_CCOPTIONS=" -fsigned-char "SEXTRA_CCOPTIONS"
```

Step 3 Run the following command to perform verification:

```
ng4ex xy01n -W png
```
If the ncl main program reports an error, as shown in the following figure, recompile the main program.

![NCL Example](image)

**Step 4** Run the following commands to download the RPM source code package of ncl:

```bash
yum install -y yum-utils
yumdownloader --source ncl
```

The source code package is `ncl-6.3.0-6.el7.src.rpm`.

**Step 5** Run the following commands to recompile ncl and decompress the ncl software package:

```bash
rpm -ivh ncl-6.3.0-6.el7.src.rpm
cd ~/rpmbuild/SOURCES
tar zxvf ncl_ncarg-6.3.0.tar.gz
```

**Step 6** Run the following commands to edit the files:

1. Run the following command to edit the `/libncarg_c/yMakefile` file:
   ```bash
   vim ncl_ncarg-6.3.0/common/src/libncarg_c/yMakefile
   ```

2. Press i to go to the edit mode and modify line 35 as follows:
   ```c
   #elif (defined(ia64) || defined(x86_64) || defined(amd64) || defined(aarch64))
   ```

3. Press Esc, type `:wq!`, and press Enter to save the file and exit.

4. Run the following command to edit the `/nio/yMakefile` file:
   ```bash
   vim ncl_ncarg-6.3.0/ni/src/lib/nio/yMakefile
   ```

5. Press i to go to the edit mode and modify line 85 as follows:
   ```c
   #if (defined(IRIX64) || defined(x86_64) || defined(ia64) || defined(__LP64__) || defined(aarch64))
   ```

6. Press Esc, type `:wq!`, and press Enter to save the file and exit.

7. Run the following command to edit the `/ncl/yMakefile` file:
   ```bash
   vim ncl_ncarg-6.3.0/ni/src/ncl/yMakefile
   ```

8. Press i to enter the editing mode.

   - Modify line 46:
     ```c
     EXTRA_CCOPTIONS = -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -fsigned-char
     ```

   - Modify line 113:
     ```c
     #if defined(IRIX64) || defined(x86_64) || defined(ia64) || defined(__LP64__) || defined(aarch64)
     ```

9. Press Esc, type `:wq!`, and press Enter to save the file and exit.
Step 7  Run the following commands to delete the original .tar packages and package the updated source code:

```
rm -f ncl_ncarg-6.3.0.tar.gz
tar zcvf ncl_ncarg-6.3.0.tar.gz ncl_ncarg-6.3.0
rm -rf ncl_ncarg-6.3.0
```

Step 8  Run the following commands to recompile ncl.

1. Run the following command to go to the SPECS directory:
   ```
cd ~/rpmbuild/SPECS
```
2. Run the following command to modify the ncl.spec file:
   ```
vim ncl.spec
```
3. Press i to go to the edit mode, and append the following content to the file:
   ```
%define _debug_install_post
   [%_rpmconfigdir]/find-debuginfo.sh %(_find_debuginfo_opts) "%(?_builddir)/%(?buildsubdir)"\%
   %nil
```
   See in the following figure:

```
* Tue Nov 8 2007 - Orion Novak <orion@cora.mera.com> - 5.0.0-1
  - Initial ncl package, based on ncarg
%define _debug_install_post
  [%_rpmconfigdir]/find-debuginfo.sh %(_find_debuginfo_opts) "%(?_builddir)/%(?buildsubdir)"\%
%nil
```

4. Press Esc, type :wq!, and press Enter to save the file and exit.
5. Run the following command to perform packaging and compilation.
   ```
rpmbuild -ba ncl.spec
```
   If an error is reported indicating that dependencies are missing, install Yum in sequence when prompted, as shown in the following figure:
   ```
yum install netcdf-fortran-devel hdf-static hdf-devel g2clib-static gdal-devel proj-devel imake libXaw-devel flex-static udunits2-devel
```

The `rpmbuild -ba ncl.spec` command can be executed when `exit 0` is displayed, as shown in the following figure:

```
error: Failed build dependencies:
netcdf-fortran-devel is needed by ncl-6.3.0-0.6.el7.aarch64
hdf-static is needed by ncl-6.3.0-0.6.el7.aarch64
devl-2.42 is needed by ncl-6.3.0-0.6.el7.aarch64
g2clib-static is needed by ncl-6.3.0-0.6.el7.aarch64
gdal-devl is needed by ncl-6.3.0-0.6.el7.aarch64
proj-devel is needed by ncl-6.3.0-0.6.el7.aarch64
imake is needed by ncl-6.3.0-0.6.el7.aarch64
libXaw-devel is needed by ncl-6.3.0-0.6.el7.aarch64
flex-static is needed by ncl-6.3.0-0.6.el7.aarch64
udunits2-devel is needed by ncl-6.3.0-0.6.el7.aarch64
```

Step 9  Run the following command to install ncl (delete the original ncl first):

```
yum erase ncl.aarch64 ncl-common.noarch ncl-devel.aarch64 ncl-examples.noarch -y
```
cd ~/rpmbuild/RPMS

cp noarch/* aarch64/

cd aarch64

yum localinstall ncl-6.3.0-6.el7.aarch64.rpm ncl-common-6.3.0-6.el7.noarch.rpm ncl-debuginfo-6.3.0-6.el7.aarch64.rpm ncl-devel-6.3.0-6.el7.aarch64.rpm ncl-examples-6.3.0-6.el7.noarch.rpm -y

----End

1.21.6 Running and Verifying ncl

Procedure

Step 1  Run the following command to perform verification:

ng4ex xy01n -W png

If the xy01n.png image is generated in the current directory, the execution is successful, as shown in Figure 1-24:

Figure 1-24 Verification result

----End
2.1 OpenFOAM 1906 Porting Guide (CentOS 7.6)

2.1.1 Introduction

Open Source Field Operation and Manipulation (OpenFOAM) is a physical field computing software developed by OpenCFD, a subsidiary of ESI. The software analyzes fluid, heat transfer, molecular dynamics, electromagnetic fluid and solid stress, and provides a visualized interface for operations from grid division to post-processing. It supports data pre-processing, post-processing, and solver customization, and is often used for computational fluid dynamics (CFD).

There are two versions of the OpenFOAM installation package provided on the official OpenFOAM website (https://openfoam.org/):

1. OpenFOAM-dev: Current development source code package of OpenFOAM, which includes the source code of the next major version of OpenFOAM.
2. OpenFOAM-7: Compilation source code on Linux distributions, which includes all latest patches. The number indicates the major version. For example, the latest major version is OpenFOAM-7.

For more information about OpenFOAM, visit the official OpenFOAM website.
Recommended Version
OpenFOAM-v1906

2.1.2 Environment Requirements

Hardware Requirements

Table 2-1 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
<tr>
<td>Network</td>
<td>1 x GE Ethernet port</td>
</tr>
<tr>
<td>Computing network</td>
<td>1 x single-port EDR/HDR IB NIC</td>
</tr>
</tbody>
</table>

OS Requirements

Table 2-2 lists the requirements on the OS.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0</td>
<td>Contained in the OS image.</td>
</tr>
</tbody>
</table>

2.1.3 Planning the Paths for Software Porting

This chapter lists the software installation paths involved in the OpenFOAM software porting.
### Table 2-3 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/path/to/OPENFOAM</td>
<td>Installation path of OpenFOAM-v1906</td>
<td>This installation path is only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

### 2.1.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

#### Configuration Process

**Table 2-4 Configuration process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>Installing ClusterShell</td>
<td>For details, see 2.1.4.1 Installing ClusterShell.</td>
</tr>
</tbody>
</table>

#### 2.1.4.1 Installing ClusterShell

**Step 1** Run the following commands to download the ClusterShell RPM installation package `clustershell-1.8.2-1.el7.noarch.rpm` and its dependency `python2-clustershell-1.8.2-1.el7.noarch.rpm`:

```bash
wget https://mirrors.tuna.tsinghua.edu.cn/epel/7/aarch64/Packages/c/clustershell-1.8.2-1.el7.noarch.rpm
wget https://mirrors.tuna.tsinghua.edu.cn/epel/7/aarch64/Packages/p/python2-clustershell-1.8.2-1.el7.noarch.rpm
```

**Step 2** Run the following commands to install ClusterShell:

```bash
yum install python-setuptools -y
```
rpm -ivh python2-clustershell-1.8.2-1.el7.noarch.rpm
clustershell-1.8.2-1.el7.noarch.rpm

NOTE

ClusterShell is installed to clear the cache of each node when multiple nodes are running. If only one node is running, you do not need to install ClusterShell.

----End

2.1.5 Obtaining the Source Code

Step 1 Download the OpenFOAM installation source code package OpenFOAM-v1906.tgz and the auxiliary third-party software installation package ThirdParty-v1906.tgz.

URLs:

Step 2 Use the SFTP tool to copy the OpenFOAM source code package and auxiliary third-party software installation package to the /path/to/OPENFOAM directory on the server.

NOTE
For details about /path/to/OPENFOAM, see Table 2-3.

----End

2.1.6 Compiling and Installing OpenFOAM

Step 1 Run the yum command to install the system dependencies:

```
yum install -y zlib-devel texinfo gstreamer-plugins-base-devel libXext-devel libGLU-devel libXt-devel libXrender-devel libXinerama-devel libpng-devel libXrandr-devel libXi-devel libXfreetype-devel libjpeg-turbo-devel libXcursor-devel readline-devel nscurses-devel mpfr-devel gmp-devel libmpc*
```

Step 2 Run the following commands to go to the directory of the installation package, and decompress the OpenFOAM source code package and third-party software installation package:

```
cd /path/to/OPENFOAM

tar zxvf OpenFOAM-v1906.tgz

tar zxvf ThirdParty-v1906.tgz
```

NOTE
For details about /path/to/OPENFOAM, see Table 2-3.

Step 3 Run the following command to edit the OpenFOAM installation configuration file:

```
vi /path/to/OPENFOAM/OpenFOAM-v1906/etc/bashrc
```

Step 4 Modify the compiler and MPI settings in the OpenFOAM installation configuration file.
export WM_COMPILER_TYPE=system
export WM_COMPILER=Gcc
export WM_MPLIB=SYSTEMOPENMPI

The three parameters indicate that the GNU compiler and OpenMPI installed in the system are used for the compilation and installation. If you need to use another compiler or MPI, modify the parameters as prompted.

**Step 5** Run the following commands to modify the compilation parameters:

```bash
sed -i 's/-O3/-O3 -march=armv8-a/g' /path/to/OPENFOAM/OpenFOAM-v1906/wmake/rules/linuxARM64Gcc/c++Opt
sed -i 's/-O3/-O3 -march=armv8-a/g' /path/to/OPENFOAM/OpenFOAM-v1906/wmake/rules/linuxARM64Gcc/cOpt
```

**Step 6** Run the following command to make the OpenFOAM configuration file `bashrc` take effect:

```bash
source /path/to/OPENFOAM/OpenFOAM-v1906/etc/bashrc
```

**NOTE**

The following is the command output:

```
No completion added for /path/to/OPENFOAM/OpenFOAM-v1906/platforms/linuxARM64GccDPInt32Opt/bin
... incorrect platform, or not yet compiled?
```

**Step 7** Run the following commands to perform compilation and installation:

```bash
cd /path/to/OPENFOAM/OpenFOAM-v1906/
./Allwmake -j 16 -s -k -q
```

**NOTE**

- `-j` indicates that all cores are used for the compilation. You can also specify the number of cores, for example, `-j 8`.
- `-s` indicates that log printing is disabled.
- `-k` indicates that errors during compilation are ignored.
- `-q` indicates that subdirectories are collected to schedule and accelerate the compilation.
- This step takes approximately two and a half hours. After the step is complete, the `bin` and `lib` directories are generated in the `/path/to/OPENFOAM/platforms/linuxARM64GccDPInt32Opt` directory. The executable files and library files required for running are available in the two directories.

---End

### 2.1.7 Running and Verifying OpenFOAM

**Step 1** Run the following commands on PuTTY to load environment variables:

```bash
export PATH=/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENMPI/lib:SLD_LIBRARY_PATH
```

**NOTE**

This step declares the MPI environment variables. If the MPI environment variables have been declared, skip this step.
Step 2 Run the following command to create the `hostfile` file and add node information:

```bash
echo -e 'node1
node2
...
nodex' > /path/to/HOSTFILE
```

**NOTE**

- In the preceding command, `node1`, `node2`, ..., and `nodex` indicate the host names of the nodes. You can run the `hostname` command to query the host names. \n indicates a newline character.
- This step is mandatory when multiple nodes are running and can be skipped when only one node is running.

Step 3 Run the following commands to load the OpenFOAM application:

```bash
Node_list=$(cat /path/to/HOSTFILE | xargs | tr ' ' ',')
clush -w ${Node_list} "cp /root/.bashrc /root/.bashrc_bak;echo source /path/to/OPENFOAM/OpenFOAM-v1906/etc/bashrc >> /root/.bashrc"
```

Step 4 Copy the case file to any location in the computing environment.

```bash
cp -r /path/to/OPENFOAM/OpenFOAM-v1906/tutorials/incompressible/pisoFoam/LES/motorBike/motorBike ./
```

Step 5 Modify the `RunFunctions` file.

```bash
vi /path/to/OPENFOAM/OpenFOAM-v1906/bin/tools/RunFunctions
```

- Multi-node:
  Add `--allow-run-as-root -x PATH -x LD_LIBRARY_PATH --hostfile /path/to/HOSTFILE` to the end of the `mpirun` command, save the modification, and exit.

  Before the modification:
  ```bash
  $mpirun -n $nProcs $appRun $appArgs "$@" </dev/null >> $logFile 2>&1
  ```

  After the modification:
  ```bash
  $mpirun --allow-run-as-root -x PATH -x LD_LIBRARY_PATH --hostfile /path/to/HOSTFILE -n $nProcs $appRun $appArgs "$@" </dev/null >> $logFile 2>&1
  ```

- Single-node:
  Add `--allow-run-as-root` to the end of the `mpirun` command, save the modification, and exit.

  Before the modification:
  ```bash
  $mpirun -n $nProcs $appRun $appArgs "$@" </dev/null >> $logFile 2>&1
  ```

  After the modification:
  ```bash
  $mpirun --allow-run-as-root -n $nProcs $appRun $appArgs "$@" </dev/null >> $logFile 2>&1
  ```

Step 6 Run the following command to modify the configuration file:

```bash
sed -ri "s/(20 8 8\))/(64 12 8\)/" motorBike/system/blockMeshDict
```

- Two-node scenario:
  ```bash
  sed -ri "s/8/256/" motorBike/system/
  {decomposeParDict,decomposeParDict.hierarchical,decomposeParDict.ptscotch}
  sed -ri "s/(4 2 1\))/(32 4 2\))/" motorBike/system/
  {decomposeParDict,decomposeParDict.hierarchical,decomposeParDict.ptscotch}
  ```
Single-node scenario:
```bash
sed -ri "s/8/128/" motorBike/system/
{decomposeParDict,decomposeParDict.hierarchical,decomposeParDict.ptscotch}
```
```bash
sed -ri "s/(4 2 1)/(16 4 2)/" motorBike/system/
{decomposeParDict,decomposeParDict.hierarchical,decomposeParDict.ptscotch}
```

**NOTE**
- If there are two nodes and 128 tasks are concurrently executed on each node, then 128 x 2 = 256, 32 x 4 x 2 = 256.
- If there is only one node and 128 tasks are concurrently executed on the node, then 128 x 1 = 128, 16 x 4 x 2 = 128.
- You can run the `lscpu |grep ^CPU(s)` command to view the number of concurrent tasks (128) on the single node. You can also change the number based on the actual configuration of the node.

**Step 7** Go to the test case directory and run the computing script.
```bash
cd motorbike
./Allrun
```

**Step 8** During the computing, the following information is displayed:
- Running blockMesh on /hpcstore/ysf/openfoam-test/1020_motorBike
- Running decomposePar on /hpcstore/ysf/openfoam-test/1020_motorBike
- Running snappyHexMesh (256 processes) on /hpcstore/ysf/openfoam-test/1020_motorBike
- Restore 0/ from 0.orig/ for processor directories
- Running renumberMesh (256 processes) on /hpcstore/ysf/openfoam-test/1020_motorBike
- Running potentialFoam (256 processes) on /hpcstore/ysf/openfoam-test/1020_motorBike
- Running checkMesh (256 processes) on /hpcstore/ysf/openfoam-test/1020_motorBike
- Running simpleFoam (256 processes) on /hpcstore/ysf/openfoam-test/1020_motorBike

**Step 9** After the computing is complete and the software exits normally, a computing log file is generated. Run the `vi log.simpleFoam` command to open the `log.simpleFoam` file.
```
```
<table>
<thead>
<tr>
<th>forceCoeffs</th>
<th>Coefficients</th>
<th>execute:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cd</strong></td>
<td>0.413353</td>
<td>(pressure: 0.398578  viscous: 0.0147753)</td>
</tr>
<tr>
<td><strong>Cs</strong></td>
<td>0.00100002</td>
<td>(pressure: 0.00999416  viscous: 6.01357e-06)</td>
</tr>
<tr>
<td><strong>Cl</strong></td>
<td>0.0683827</td>
<td>(pressure: 0.0683764  viscous: 6.2544e-06)</td>
</tr>
<tr>
<td><strong>CmRoll</strong></td>
<td>-0.0101679</td>
<td>(pressure: -0.0102318  viscous: 6.39234e-05)</td>
</tr>
<tr>
<td><strong>CmPitch</strong></td>
<td>0.15315</td>
<td>(pressure: 0.146948  viscous: 0.00620151)</td>
</tr>
<tr>
<td><strong>CmYaw</strong></td>
<td>0.0118663</td>
<td>(pressure: 0.0120074  viscous: -0.000141087)</td>
</tr>
<tr>
<td><strong>Cd(f)</strong></td>
<td>0.196509</td>
<td></td>
</tr>
<tr>
<td><strong>Cd(r)</strong></td>
<td>0.216844</td>
<td></td>
</tr>
<tr>
<td><strong>Cs(f)</strong></td>
<td>0.0168664</td>
<td></td>
</tr>
<tr>
<td><strong>Cs(r)</strong></td>
<td>-0.00686619</td>
<td></td>
</tr>
<tr>
<td><strong>Cl(f)</strong></td>
<td>0.187341</td>
<td></td>
</tr>
<tr>
<td><strong>Cl(r)</strong></td>
<td>-0.118959</td>
<td></td>
</tr>
</tbody>
</table>
```
```
anskightWrite ensightWrite write: ( k omega p U )
End
```
```
Finalising parallel run
```

-----End
2.1.8 Troubleshooting

Problem 1: "fatal error: boost/config/compiler/gcc.hpp: No such file or directory" Is Displayed During Compilation

Symptom
When Allwmake is run to compile the Computational Geometry Algorithms Library (CGAL), the compilation fails and "fatal error: boost/config/compiler/gcc.hpp: No such file or directory" is displayed.

Possible Cause
The CGAL fails to be installed due to environment variable or CMake installation failure. If the error occurs during the installation of a module of OpenFOAM, you only need to install the module again.

Procedure

Step 1  Delete the installed executable file and path.

```
rm -rf /path/to/OPENFOAM/OPENFOAM-v1906/platform/arch/boost_1_64_0
```

Step 2  Run the independent installation script in Thirdparty-v1906.

```
./makeCGAL
```

----End

Problem 2: "esmump.h* No such file or directory" Is Displayed During Compilation

Symptom
When Allwmake is run to compile Scotch, the compilation fails and "esmump.h* No such file or directory" is displayed.

Possible Cause
An error occurs during script running. As a result, esmump fails to be installed.

Procedure

Step 1  Go to the Scotch directory and manually compile esmumps.

```
cd /path/to/OPENFOAM/THirdParty-v1906/scotch_6.0.6/src
make esmumps
```

Step 2  Run the Allwmake script again to continue the compilation and installation.

```
./Allwmake -j
```

----End
Problem 3: " cannot stat `../bin/d[agm]***" Is Displayed During Compilation

Symptom
When Allwmake is run to compile Scotch, the compilation fails and the following message is displayed:

cp: cannot stat `../bin/d[agm]**': No such file or directory
make: [install] Error 1 (ignored)

Possible Cause
This problem is not caused by d[agm]. Generally, this problem occurs when the Scotch directory is not cleared before the installation program is rerun. You can run the make distclean command in the Scotch source code directory and then manually install the Scotch program.

Procedure

Step 1 Delete the directory generated by the failed installation.

**NOTE**

- linuxARM64GccDPInt32 is named based on the installation environment architecture and compiler. You need to change it to the actual path.

```
rm -rf /path/to/OPENFOAM/ThirdParty-v1906/platforms/linuxARM64GccDPInt32/scotch_6.0.6/
```

Step 2 Go to the Scotch source-code directory and manually compile and install the software.

**NOTE**

To manually install Scotch, you need to set the installation path to the installation path that is deleted previously.

```
make scotch
make esmumps
make ptscotch
make prefix=/path/to/OPENFOAM/ThirdParty-v1906/platforms/linuxARM64GccDPInt32/scotch_6.0.6/ install
```

Step 3 Run the Allwmake script again to continue the compilation and installation.

```
foam
./Allwmake -j
```

Problem 4: The Case Cannot Stop Properly and "ls: cannot access processor*: No such file or directory" Is Displayed

Symptom
Generally, case running takes several minutes. If the case is stopped within one minute, open the log file to check whether the computing is running properly. If
the log file contains the error information, "mpirun has detected an attempt to run as root." is displayed.

Possible Cause
When running the mpirun command as the root user, you need to add a parameter.

Procedure

Step 1 Run the vi /path/to/OPENFOAM/OpenFOAM-v1906/bin/tools/RunFunctions command.

Step 2 Add --allow-run-as-root to the end of the mpirun command, save the modification, and run the case again.

./Allclean

./Allrun

----End

2.2 SU2 Porting Guide (CentOS 7.6)

2.2.1 Introduction

SU2 is a high-precision partial differential equation solver developed by the Department of Aeronautics and Astronautics of Stanford University. Different from CFL3D of NASA, SU2 is an unstructured grid solver developed based on C++ (similar to OpenFOAM). It carries out not only serial computing but also parallel computing, and has the function of deformation grid (that is, dynamic grid) and grid adaptive function. To improve the computing efficiency, the code adopts multiple grids to accelerate the convergence.

SU2 has multiple spatial discrete formats and multiple flux limiters, which can be used to calculate the flow rate from low speed to high speed, it can also simulate multiple physical fields, multicomponent flow, combustion, aerodynamic noise, two-phase flow, magnetohydrodynamic, plasma flow, and optimization.

For more information, visit the official SU2 website.

Programming language: C++

Brief description: a high-precision partial differential equation solver

Open-source license: GPL 2.1

2.2.2 Environment Requirements

Hardware Requirements

Table 2-5 lists the hardware requirements.
Table 2-5 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 2-6 lists the software requirements.

Table 2-6 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>SU2</td>
<td>6.0.1</td>
<td><a href="https://su2code.github.io/">https://su2code.github.io/</a></td>
</tr>
<tr>
<td></td>
<td>7.0.4</td>
<td><a href="https://github.com/su2code/SU2/archive/v7.0.4.tar.gz">https://github.com/su2code/SU2/archive/v7.0.4.tar.gz</a></td>
</tr>
<tr>
<td>HDF5 (optional)</td>
<td>1.10.5</td>
<td><a href="https://www.hdfgroup.org/downloads/hdf5/source-code/">https://www.hdfgroup.org/downloads/hdf5/source-code/</a></td>
</tr>
<tr>
<td>Test case</td>
<td>Inviscid_Bump</td>
<td><a href="https://github.com/su2code/su2code.github.io">https://github.com/su2code/su2code.github.io</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 2-7 lists the OS requirements.

Table 2-7 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

2.2.3 Planning the Paths for Software Porting

Table 2-8 lists the software installation paths involved in the SU2 software porting.
Table 2-8 Paths for software porting

<table>
<thead>
<tr>
<th>N o.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/SU2</td>
<td>Installation path of SU2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/CASE</td>
<td>Path for storing the SU2 test case</td>
<td></td>
</tr>
</tbody>
</table>

2.2.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 2-9 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install HDF5.</td>
<td>For details, see 2.2.4.1 Installing HDF5.</td>
</tr>
</tbody>
</table>

2.2.4.1 Installing HDF5

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the HDF5 installation package:

```
tar xzvf hdf5-1.10.5.tar.gz
```
Step 3  Run the following command to switch to the directory generated after the package is decompressed:

```
cd hdf5-1.10.5
```

**Step 4**  Run the following command perform configuration:

```
CC=mpicc ./configure --prefix=/path/to/HDF5/ --enable-parallel
```

**Step 5**  Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

**Step 6**  Run the following command to configure the compilation environment:

```
export HDF5_ROOT=/path/to/HDF5/
```

----End

### 2.2.5 SU2 6.0.1 Porting Guide (CentOS 7.6)

#### 2.2.5.1 Obtaining the Source Code

**Procedure**

**Step 1**  Download the SU2 installation package **SU2-6.0.1.tar.gz**.

URL: [https://su2code.github.io/](https://su2code.github.io/)

**Step 2**  Use SFTP to upload the SU2 installation package to the `/path/to/SU2` directory on the server.

----End

#### 2.2.5.2 Compiling and Installing SU2

**Procedure**

**Step 1**  Use PuTTY to log in to the server as the `root` user.

**Step 2**  Run the following command to install the dependency:

```
yum install numpy.aarch64 -y
```

**Step 3**  Run the following commands to decompress the SU2 installation package:

```
tar zxvf SU2-6.0.1.tar.gz
```

**Step 4**  Run the following command to switch to the directory generated after the package is decompressed:

```
cd SU2-6.0.1
```

**Step 5**  Run the following command to run the `bootstrap` script:

```
./bootstrap
```
Step 6  Run the following command to switch to the SU2 installation directory:

```
cd /path/to/SU2/
```

Step 7  Run the following command perform configuration:

```
SU2-6.0.1/configure --prefix=/path/to/SU2/ CXXFLAGS="-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110" --enable-mpi --with-cc=mpicc --with-cxx=mpicxx --with-HDF5-lib=/path/to/HDF5/lib --with-HDF5-include=/path/to/HDF5/include
```

Step 8  Run the following commands to perform compilation and installation:

```
make -j 32
make install
```

Step 9  Run the following commands to add the environment variables:

```
export SU2_RUN=/path/to/SU2/bin
export SU2_HOME=/path/to/SU2
export PATH=$PATH:$SU2_RUN
export PYTHONPATH=$PYTHONPATH:$SU2_RUN
```

---End

## 2.2.6 SU2 7.0.4 Porting Guide (CentOS 7.6)

### 2.2.6.1 Obtaining the Source Code

**Procedure**

**Step 1**  Download the SU2 installation package SU2-7.0.4.tar.gz.

Download address: [https://github.com/su2code/SU2/archive/v7.0.4.tar.gz](https://github.com/su2code/SU2/archive/v7.0.4.tar.gz)

**Step 2**  Use the SFTP tool to upload the SU2 installation package to the /path/to/SU2 directory on the server.

---End

### 2.2.6.2 Compiling and Installing SU2

**Procedure**

**Step 1**  Use PuTTY to log in to the server as the root user.

**Step 2**  Run the following command to install the dependency:

```
yum install numpy.aarch64 -y
```

**Step 3**  Run the following command to decompress the SU2 installation package:

```
tar zxfv SU2-7.0.4.tar.gz
```
Step 4 Run the following command to switch to the directory generated after decompression:
```bash
cd SU2-7.0.4
```

Step 5 Execute the `bootstrap` script.
```bash
./bootstrap
```

Step 6 Run the following commands to switch to the SU2 installation directory:
```bash
mkdir ../SU2-7.0.4-gnu-9.3.build
cd ../SU2-7.0.4-gnu-9.3.build
```

Step 7 Run the following command to perform configuration:
```bash
../SU2-7.0.4/configure --prefix=/path/to/SU2/ CXXFLAGS="-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110" --enable-mpi --with-cc=mpicc --with-cxx=mpicxx
```

Step 8 Run the following commands to compile and install SU2:
```bash
make -j 32
make install
```

Step 9 Run the following commands to add the environment variables:
```bash
export SU2_RUN=/path/to/SU2/bin
export SU2_HOME=/path/to/SU2
export PATH=$PATH:$SU2_RUN
export PYTHONPATH=$PYTHONPATH:$SU2_RUN
```

---End

2.2.7 Running and Verifying SU2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Download the test case package `su2code.github.io-master.zip`.

---NOTICE---
The parameters in the case configuration files of SU2 6.X.X and SU2 7.X.X are greatly changed. You need to obtain the case configuration file of the corresponding version from the corresponding source code package.

---END---

Step 3 Use SFTP to upload the SU2 test case package to the `/path/to/CASE` directory on the server.

Step 4 Run the following command to decompress the test case package:
```
unzip su2code.github.io-master.zip
```
Step 5 Run the following command to switch to the directory generated after the package is decompressed:

```
cd su2code.github.io-master
```

Step 6 Run the following command to create the `testcase` folder:

```
mkdir testcase
```

Step 7 Run the following commands to copy the `testcase` folder and rename it `Inviscid_Bump`:

```
cd testcase
cp -rf /path/to/CASE/su2code.github.io-master/testcase /Inviscid_Bump
```

Step 8 Run the following command to go to the `Inviscid_Bump` test case package:

```
cd ../Inviscid_Bump
```

Step 9 Run the following command to start SU2:

```
mpirun --allow-run-as-root -np 16 SU2_CFD inv_channel.cfg
```

Information similar to the following is displayed.

```
----------------------- Residual Evolution Summary ----------------------
log10[Maximum residual]: -7.391.
Maximum residual point 511, located at (3, 0.003).
-------------------------------------------------------------------------
Iter    Time(s)     Res[Rho]     Res[RhoE]      CL(Total)      CD(Total)
600   0.087015    -9.260167     -3.862403       0.000291       0.000007
601   0.087013    -9.271532     -3.873152       0.000291       0.000007
602   0.087011    -9.282237     -3.883903       0.000291       0.000007
603   0.087010    -9.293729     -3.894743       0.000291       0.000007
604   0.087008    -9.305008     -3.905581       0.000291       0.000007
605   0.087006    -9.316490     -3.916410       0.000291       0.000007
606   0.087006    -9.327466     -3.927381       0.000291       0.000007
607   0.087004    -9.338974     -3.938270       0.000291       0.000007
608   0.087002    -9.350370     -3.949267       0.000291       0.000007
609   0.087000    -9.361588     -3.960211       0.000291       0.000007
610   0.086998    -9.373061     -3.971327       0.000291       0.000007
611   0.086997    -9.384581     -3.982358       0.000291       0.000007
-------------------------------------------------------------------------
-------------------------- File Output Summary --------------------------
Writing comma-separated values (CSV) surface files.
Loading solution output data locally on each rank.
Sorting output data across all ranks.
Writing binary SU2 native restart file.
-------------------------------------------------------------------------
------------------------- Solver Postprocessing -------------------------
Deleted CNumerics container.
Deleted CIntegration container.
Deleted CSolver container.
Deleted CIteration container.
Deleted CInterpolator container.
Deleted CTransfer container.
Deleted CGeometry container.
Deleted CFreeFormDefBox class.
Deleted CSurfaceMovement class.
Deleted CVolumetricMovement class.
Deleted CConfig container.
Deleted COutput class.
-------------------------------------------------------------------------
Completed in 53.260358 seconds on 16 cores.
-------------------------------------------------- Exit Success (SU2_CFD) -----------------------------------------------

----End
2.2.8 Troubleshooting

Problem 1: An Error Is Reported During SU2 6.0.1 Compilation

Symptom
An error message "error:'VALID_REF' was not declared in this scope; did you mean'INVALID_REF'" is displayed when SU2 is compiled.

Possible Cause
This error is caused by a code bug.

Procedure
Download the official release of 6.0.x.

Problem 2: An Error Is Reported During SU2 7.0.4 Compilation

Symptom
An error message "configure: error: cannot find python-config for /usr/bin/python." is displayed when SU2 is compiled.

Possible Cause
The dependency python-devel does not exist.

Procedure
Run the following command to install the dependency before configuring:

yum install python-devel -y

2.2.9 More Information

Official SU2 website:
https://su2code.github.io/

SU2 usage guide:
https://su2code.github.io/docs/Execution/#collapse-3

2.3 PALABOS 2.1r0 Porting Guide (CentOS 7.6)

2.3.1 Introduction

The Palabos library is a common computational fluid dynamics (CFD) framework. Its kernel is based on the lattice Boltzmann (LB) method. It can be used as a research tool or an engineering tool. It provides a friendly programming interface and can be easily used to simulate fluid flow.

For more information about Palabos, visit the official PALABOS website.

Language: C++
Brief description: Palabos is fluid dynamics simulation software.
Open-source protocol: GPL 3.0

**Recommended Tool Version**

Palabos v2.1r0

**2.3.2 Environment Requirements**

**Hardware Requirements**

Table 2-10 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

**Software requirements**

Table 2-11 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>PALABOS</td>
<td>v2.1r0</td>
<td><a href="https://gitlab.com/unigespc/palabos/-/tree/v2.1r0">https://gitlab.com/unigespc/palabos/-/tree/v2.1r0</a></td>
</tr>
<tr>
<td>ImageMagick</td>
<td>6.7.8.9-15.el7_2</td>
<td>RPM package contained in the system image</td>
</tr>
<tr>
<td>Python</td>
<td>3.6.3</td>
<td><a href="https://www.python.org/downloads/source/">https://www.python.org/downloads/source/</a></td>
</tr>
<tr>
<td>Test case</td>
<td>cavity3d</td>
<td>Benchmark case provided by the software</td>
</tr>
</tbody>
</table>

**OS Requirements**

Table 2-12 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
2.3.3 Planning the Paths for Software Porting

Table 2-13 lists the software installation paths involved in the Palabos software porting.

Table 2-13  Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each installation package for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>4</td>
<td>/path/to/ PALABOS</td>
<td>Installation path of Palabos</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>5</td>
<td>/path/to/ PYTHON3</td>
<td>Installation path of Python 3</td>
<td></td>
</tr>
</tbody>
</table>

2.3.4 Configuring the Compilation Environment

Prerequisites

Use SFTP to upload the installation packages to the planned directories on the server.

Configuration process

Table 2-14 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Environment for the Cluster Scenario&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing Python 3</td>
<td>For details, see Procedure.</td>
</tr>
<tr>
<td>3</td>
<td>Installing ImageMagick</td>
<td>For details, see 2.3.4.2 Installing ImageMagick.</td>
</tr>
</tbody>
</table>
2.3.4.1 Installing Python 3

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to install the dependencies required by Python:

```
yum -y install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel gdbm-devel db4-devel libpcap-devel xz-devel
```

**Step 3** Run the following commands to decompress the Python 3 installation package:

```
tar -xvf Python-3.6.3.tgz
```

**Step 4** Run the following command to switch to the directory containing the decompressed files:

```
cd Python-3.6.3
```

**Step 5** Run the following command perform configuration:

```
./configure --prefix=/path/to/PYTHON3
```

**Step 6** Run the following commands to perform compilation and installation:

```
make -j
make install
```

**Step 7** Run the following commands to set the Python 3 environment variables:

```
export PATH=/path/to/PYTHON3/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON3/lib:$LD_LIBRARY_PATH
```

----End

2.3.4.2 Installing ImageMagick

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the ImageMagick installation package:

```
yum install ImageMagick-6.7.8.9-15.el7_2.aarch64 -y
```

----End

2.3.5 Obtaining the Source Code

Procedure

**Step 1** Download the Palabos installation package **palabos-v2.1r0.zip**.

Download address: [https://gitlab.com/unigespc/palabos/-/tree/v2.1r0](https://gitlab.com/unigespc/palabos/-/tree/v2.1r0)
Step 2 Use SFTP to upload the Palabos installation package to the /path/to/PALABOS directory on the server.

2.3.6 Compiling and Installing the Software

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following command to create a main program installation directory:
```bash
mkdir -p /path/to/PALABOS
```
Step 3 Copy the installation package to the installation directory.
```bash
cp palabos-v2.1r0.zip /path/to/PALABOS
```
Step 4 Run the following command to go to the program installation directory:
```bash
cd /path/to/PALABOS
```
Step 5 Run the following commands to decompress the installation package:
```bash
unzip palabos-v2.1r0.zip
```
Step 6 Run the following command to go to the cavity2d case directory:
```bash
cd /path/to/PALABOS/palabos-v2.1r0/examples/showCases/cavity2d
```
Step 7 Compile and install the program according to the case.
```bash
make -j 16
```
After the installation is successful, run the `ll` command to view the generated cavity2d binary executable file in the current directory.

NOTE
Palabos builds a binary executable file based on the case model. The following uses the cavity2d case file as an example.

2.3.7 Running and Verifying the Software

2.3.7.1 Running and Verifying the cavity2d Case

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following command to set the environment variables:
```bash
export PATH=/path/to/GNU/bin: /path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
SLD_LIBRARY_PATH
```
Step 3 Run the following command to create the hostfile file and add node information:

```
echo -e 'node1
node2
...
nodex' > /path/to/HOSTFILE
```

**NOTE**

In the preceding command, `node1`, `node2`, ..., and `nodex` indicate the host names of the nodes. You can run the `hostname` command to query the host names. `\n` indicates a newline character.

This step is mandatory when multiple nodes are running and can be skipped when only one node is running.

Step 4 Run the following command to go to the cavity2d case directory:

```
cd /path/to/PALABOS/palabos-v2.1r0/examples/showCases/cavity2d
```

Step 5 Run the following command to run the test case:

```
time mpirun --allow-run-as-root -mca btl ^openib --hostfile /path/to/HOSTFILE -x OMP_NUM_THREADS=1 -x PATH -x LD_LIBRARY_PATH -N 96 ./cavity2d
```

**NOTE**

Run the following command on a single node to run the test case:

```
time mpirun --allow-run-as-root -mca btl ^openib -np 96 ./cavity2d
```

After the case is run, you can run the `ll` command to view the generated GIF image file in the `tmp` directory.

-----End

### 2.3.7.2 Running and Verifying the boussinesqThermal3d Case

#### Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to set the environment variables:

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:SPATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
SDL_LIBRARY_PATH
```

**Step 3** Run the following command to create the hostfile file and add node information:

```
echo -e 'node1
node2
...
nodex' > /path/to/HOSTFILE
```

**NOTE**

- In the preceding command, `node1`, `node2`, ..., and `nodex` indicate the host names of the nodes. You can run the `hostname` command to query the host names. `\n` indicates a newline character.
- This step is mandatory when multiple nodes are running and can be skipped when only one node is running.

**Step 4** Run the following command to go to the cavity2d case directory:
Step 5 Run the following commands to perform compilation and installation:

```bash
cd /path/to/PALABOS/palabos-v2.1r0/examples/showCases/boussinesqThermal3d
make -j 16
```

Step 6 Run the following command to run the test case:

```bash
time mpirun --allow-run-as-root -mca btl ^openib --hostfile /path/to/HOSTFILE -x OMP_NUM_THREADS=1 -x PATH -x LD_LIBRARY_PATH -N 96 ./rayleighBenard3D 100
```

**NOTE**

Run the following command on a single node to run the test case:

```bash
time mpirun --allow-run-as-root -mca btl ^openib -np 96 ./rayleighBenard3D 100
```

Step 7 After the case is run, you can run the `ll` command to view the generated VTI file in the `tmp` directory.

The following is an example of the command output.

![Figure 2-1 Result example](image)

**NOTE**

In the preceding figure, **VTK** is the VTI file. The file name extension is not displayed.

----End

### 2.3.8 Support and Other Resources

Official Palabos website:

*https://palabos.unige.ch/*

PALABOS installation guide:

*https://gitlab.com/unigespc/palabos/-/tree/v2.1r0*

### 2.4 deal.II 9.1.1 Porting Guide (CentOS 7.6)
2.4.1 Introduction

deal.II was originally meant to be a successor to the differential equation analysis library, a C++ library designed to use the adaptive finite element method for partial differential equation calculations. It uses state-of-the-art programming techniques to provide you with modern interfaces to complex data structures and algorithms that you need.

deal.II aims to quickly develop modern finite element code by using various tools frequently used in adaptive grids and finite element programs. Writing such procedures is not easy, and successful programs tend to become very large and complex. You are recommended to do this with a library that handles the details of grid processing and refinement, degree of freedom, grid input, and output of graphical format results. Similarly, the way that multiple space sizes are supported at a time allows programs to be written independently of the space size and run without causing memory consumption.

For more information about deal.II, visit the official deal.II website.

Programming language: C++

Brief description: an open source finite element software program used to solve partial differential equations

Open-source license: GNU Lesser General Public version 2.1

Recommended Version

deal.II 9.1.1

2.4.2 Environment Requirements

Hardware Requirements

Table 2-15 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 2-16 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>deal.II</td>
<td>9.1.1</td>
<td><a href="https://www.dealii.org/download.html">https://www.dealii.org/download.html</a></td>
</tr>
</tbody>
</table>
OS Requirements

Table 2-17 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

2.4.3 Planning the Paths for Software Porting

Table 2-18 describes the software installation paths involved in the deal.II software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/DEAL.II</td>
<td>Installation path of deal.II</td>
<td>The installation path provided here is only an example. A shared path is recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

2.4.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.
### Procedure

#### Table 2-19 Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Environment for the Cluster Scenario&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
</tbody>
</table>

### 2.4.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the `dealii-9.1.1.tar.gz` installation package.

URL: [https://www.dealii.org/download.html](https://www.dealii.org/download.html)

**Step 2** Use the SFTP tool to upload the deal.II installation package to the `/path/to/DEAL.II` directory on the server.

---End

### 2.4.6 Compiling and Installing deal.II

**Procedure**

**Step 1** Use PuTTY to log in to the server as the `root` user.

**Step 2** Run the following command to install the dependency:

```bash
yum install blas lapack zlib boost -y
```

**Step 3** Run the following command to decompress the deal.II installation package:

```bash
tar -xzvf dealii-9.1.1.tar.gz
```

**Step 4** Run the following command to switch to the directory generated after decompression:

```bash
cd dealii-9.1.1
```

**Step 5** Run the following commands to create and access the directory for compiling deal.II:

```bash
mkdir build
cd build
```

**Step 6** Run the following commands to perform compilation and installation:

```bash
CC=mpicc CXX=mpicxx FC=mf90 cmake -DDEAL_II_WITH_THREADS=OFF -DDEAL_II_COMPONENT_DOCUMENTATION=ON -DDEAL_II_WITH_MPI=ON -DCMAKE_INSTALL_PREFIX=/path/to/DEAL.II ../
```
Step 7  Run the following command to modify the `build.make` file:

1. vi `tests/quick_tests/CMakeFiles/mpi.debug.run.dir/build.make`
2. Press `i` to enter the editing mode and add the content in bold to line 52 in the `build.make` file.

   ```
   ```

3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

Step 8  Run the following commands to check the setting:

```
make -j 32
make install
make test
```

If 100% tests passed is displayed, the program is successfully deployed, as shown in Figure 2-2.

Figure 2-2 Result example

[Running quicktests...]

```text
100% tests passed, 0 tests failed out of 6
Total test time (real) = 24.08 sec
Built targets test

----End
```

2.4.7 Running and Verifying deal.II

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to switch to the directory where the test case is stored:

```
cd /path/to/deal.II/examples/step-23
```

Step 3  Run the following commands to compile the step-23 test case:

```
cmake .
make
```

Step 4  Run the following command to create the `hostfile` file and add node information:
Step 5 Run the following command to perform the step-23 test case:
- Single-node:
  \{ time make run ; } 2>&1 |tee -a test.out
- Multi-node:
  \{ time mpirun --allow-run-as-root --hostfile /path/to/HOSTFILE -x OMP_NUM_THREADS=1 -x PATH -x LD_LIBRARY_PATH -N 2 ./step-23 ; } 2>&1 |tee -a test.out

Check the value of real in the test.out log. A smaller value indicates better single-core performance.

Figure 2-3 is an example of the output result.

2.4.8 More Information

Official deal.II installation guide:
https://www.dealii.org/9.1.1/readme.html#optional

2.5 AMG 1.2 Porting Guide (CentOS 7.6)

2.5.1 Introduction

Algebraic Multigrid Benchmark (AMG) is a parallel algebraic multigrid solver for linear systems arising from problems on unstructured grids. AMG is written in ISO-C. It is an SPMD code which uses MPI and OpenMP threading within MPI tasks. Parallelism is achieved by data decomposition. The driver provided with AMG achieves this decomposition by simply subdividing driver provided with AMG
achieves this decomposition by simply subdividing the grid into logical $P \times Q \times R$ (in 3D) chunks of equal size.

For details about AMG, visit the official AMG website.

Language: C

Brief description: A parallel algebraic multigrid solver for linear systems arising from problems on unstructured grids.

Open-source license: GPL

**Recommended Version**

AMG AMG-1.2

### 2.5.2 Environment Requirements

**Hardware Requirements**

*Table 2-20* lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

**Software Requirements**

*Table 2-21* lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMG</td>
<td>AMG-1.2</td>
<td><a href="https://github.com/LLNL/AMG/tree/1.2">https://github.com/LLNL/AMG/tree/1.2</a></td>
</tr>
</tbody>
</table>

**OS Requirements**

*Table 2-22* lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
2.5.3 Planning the Paths for Software Porting

Table 2-23 lists the software installation paths involved in the AMG software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/AMG</td>
<td>Installation path of AMG</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

2.5.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Procedure

Table 2-24 Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>
2.5.5 Obtaining the Source Code

Procedure

Step 1 Download the AMG installation package AMG-1.2.zip.
URL: https://github.com/LLNL/AMG/tree/1.2

Step 2 Use SFTP to upload the AMG installation package to the /path/to/AMG directory on the server.

----End

2.5.6 Compiling and Installing AMG

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the AMG installation package:

```bash
unzip AMG-1.2.zip
```

Step 3 Run the following command to switch to the directory generated after decompression:

```bash
cd /path/to/AMG/AMG-1.2
```

Step 4 Run the following commands to perform compilation:

```bash
make
```

**NOTE**
In Makefile.include, -DHYPRE_USING_PERSISTENT_COMM optimizes the MPI and -DHYPRE_HOPSCOTCH optimizes the OpenMP.

----End

2.5.7 Running and Verifying AMG

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to create the hostfile file and add node information:

```bash
echo -e 'node1
node2
...nnodex' > /path/to/HOSTFILE
```

**NOTE**
- In the preceding command, node1, node2, ..., and nodex indicate the host names of the nodes. You can run the hostname command to query the host names. \n indicates a newline character.
- This step is mandatory when multiple nodes are running and can be skipped when only one node is running.
Step 3  Run the following command to go to the test directory. After the compilation, the amg binary file is generated.

```bash
cd /path/to/AMG/AMG-1.2/test
```

Step 4  Run the following command to run the amg binary file:

```bash
mpirun --allow-run-as-root -mca btl ^openib --hostfile /path/to/HOSTFILE -x OMP_NUM_THREADS=1 -x PATH -x LD_LIBRARY_PATH -N 96 amg -P 4 6 4 -n 100 100 100 -printstats
```

**NOTE**

1. `-p <Px> <Py> <Pz>`: defines the processor topology for each part. The total number of MPI processes is Px * Py * Pz.
2. `-n <nx> <ny> <nz>`: defines the size of each processor for problems of the multi-dimensional data set. The `-n` option allows each MPI process to specify the size of a local problem, generating the global problem size of <Pz> * <nz> for <Px> * <nx> and <Py> * <ny>.
3. `-printstats`: displays detailed information about the AMG preprocessor and the number of iterations.
4. `-N`: indicates the number of tasks on each node. The value of Px * Py * Pz must be equal to the value of -N multiplied by the number of nodes.
5. If only one node is running, run the following command:

```bash
mpirun --allow-run-as-root -mca btl ^openib -x OMP_NUM_THREADS=1 -np 96 amg -P 4 6 4 -n 100 100 100 -printstats
```

The following is an example of the command output:

```
BoomerAMG SOLVER PARAMETERS:
Maximum number of cycles:         1
Stopping Tolerance:               0.000000e+00
Cycle type (1 = V, 2 = W, etc.):  1
Relaxation Parameters:
Visiting Grid:                     down   up  coarse
   Number of sweeps:            2    2     1
Type 0=Jac, 3=hGS, 6=hSGS, 9=GE:     18   18     9
Point types, partial sweeps (1=C, -1=F):
   Pre-CG relaxation (down):   0   0
   Post-CG relaxation (up):   0   0
   Coarsest grid:   0
=============================================  
Problem 1: AMG Setup Time:
========================================================================
PCG Setup:
   wall clock time = 6.333697 seconds
   wall MFLOPS =  0.000000
   cpu clock time = 6.350000 seconds
   cpu MFLOPS =  0.000000
FOM_Setup: nnz_AP / Setup Phase Time: 4.452531e+08
========================================================================
Problem 1: AMG-PCG Solve Time:
========================================================================
PCG Solve:
   wall clock time = 34.855871 seconds
   wall MFLOPS =  0.000000
   cpu clock time = 34.860000 seconds
   cpu MFLOPS =  0.000000
   Iterations = 23
   Final Relative Residual Norm = 6.624473e-09
FOM_Solve: nnz_AP * Iterations / Solve Phase Time: 1.860870e+09
Figure of Merit (FOM_1):  1.506966e+09

----End
```
2.6 Code_Saturne 6.0.2 Porting Guide (CentOS 7.6)

2.6.1 Introduction

Code_Saturne is the free, open-source software developed and released by EDF to solve computational fluid dynamics (CFD) applications.

It solves the Navier-Stokes equations for 2D, 2D-axisymmetric and 3D flows, steady or unsteady, laminar or turbulent, incompressible or weakly dilatable, isothermal or not.

For details about Code_Saturne, visit the official Code_Saturne website.

Language: C

Brief description: free, open-source CFD software developed and released by EDF

Open-source license: GNU GPL licence

Recommended Version

Code_Saturne 6.0.2

2.6.2 Environment Requirements

Hardware Requirements

Table 2-25 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 2-26 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code_Saturne</td>
<td>v6.0.2</td>
<td><a href="https://github.com/code-saturne/code_saturne/releases">https://github.com/code-saturne/code_saturne/releases</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.3</td>
<td><a href="https://support.hdfgroup.org/ftp/HDF/releases/hdf5-1.10.3/src/hdf5-1.10.3.tar.gz">https://support.hdfgroup.org/ftp/HDF/releases/hdf5-1.10.3/src/hdf5-1.10.3.tar.gz</a></td>
</tr>
</tbody>
</table>
### OS Requirements

Table 2-27 lists the OS requirements.

**Table 2-27 OS requirements**

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 2.6.3 Planning the Paths for Software Porting

Table 2-28 lists the software installation paths involved in the Code_Saturne software porting.

**Table 2-28 Paths for software porting**

<table>
<thead>
<tr>
<th>N o.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/ CODE_SATURNE</td>
<td>Installation path of Code_Saturne</td>
<td>The planned installation paths are only examples. You are advised to deploy the software in shared paths. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/ HDF5</td>
<td>Installation path of HDF5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/MED</td>
<td>Installation path of MED</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/ SCOTCH</td>
<td>Installation path of SCOTCH</td>
<td></td>
</tr>
</tbody>
</table>
### Table 2-29 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Installing the BLAS dependency</td>
<td>For details, see <a href="#">2.6.4.1 Installing the BLAS Dependency</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Installing PyQt4</td>
<td>For details, see <a href="#">2.6.4.2 Installing PyQt4</a>.</td>
</tr>
<tr>
<td>4</td>
<td>Installing HDF5</td>
<td>For details, see <a href="#">2.6.4.3 Installing HDF5</a>.</td>
</tr>
<tr>
<td>5</td>
<td>Installing MED</td>
<td>For details, see <a href="#">2.6.4.4 Installing MED</a>.</td>
</tr>
<tr>
<td>6</td>
<td>Installing SCOTCH</td>
<td>For details, see <a href="#">2.6.4.5 Installing SCOTCH</a>.</td>
</tr>
<tr>
<td>7</td>
<td>Installing METIS</td>
<td>For details, see <a href="#">2.6.4.6 Installing METIS</a>.</td>
</tr>
</tbody>
</table>

#### 2.6.4.1 Installing the BLAS Dependency

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to install the dependency:

```
yum install blas blas-devel -y
```

----End
2.6.4.2 Installing PyQt4

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install PyQt4.

```
yum install PyQt4.aarch64 PyQt4-devel.aarch64 -y
```

-----End

2.6.4.3 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the HDF5 installation package:

```
tar -zxvf hdf5-1.10.3.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd hdf5-1.10.3
```

Step 4 Run the following command to configure HDF5:

```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared
```

Step 5 Run the following commands to perform compile and install HDF5:

```
make -j
make install
```

Step 6 Run the following commands to set HDF5 environment variables:

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:SLD_LIBRARY_PATH
```

-----End

2.6.4.4 Installing MED

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the MED installation package:

```
tar -zxvf med-4.0.0.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:
cd med-4.0.0

Step 4 Run the following command to configure MED:

./configure --prefix=/path/to/MED --with-hdf5=/path/to/HDF5 --disable-python

Step 5 Run the following commands to perform compile and install MED:

make -j
make install

Step 6 Run the following commands to set the MED environment variables:

export PATH=/path/to/MED/bin:$PATH
export LD_LIBRARY_PATH=/path/to/MED/lib:$LD_LIBRARY_PATH

----End

2.6.4.5 Installing SCOTCH

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the SCOTCH installation package:

tar -zxvf scotch_6.0.9.tar.gz

Step 3 Run the following command to switch to the directory generated after decompression:

cd scotch_6.0.9

Step 4 Run the following command to create a soft link for Makefile.inc:

cd src

ln -s ./Make.inc/Makefile.inc.x86-64_pc_linux2.shlib Makefile.inc

Step 5 Run the following commands to perform compile and install SCOTCH:

make scotch

Step 6 Run the following commands to set the SCOTCH environment variables:

export PATH=/path/to/SCOTCH/bin:$PATH
export LD_LIBRARY_PATH=/path/to/SCOTCH/lib:$LD_LIBRARY_PATH

----End

2.6.4.6 Installing METIS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the METIS installation package:
tar -zxvf metis-5.1.0.tar.gz

**Step 3** Run the following command to switch to the directory generated after decompression:

cd metis-5.1.0

**Step 4** Run the following commands to compile and install METIS:

```bash
make config prefix=/path/to/METIS
make install
```

**Step 5** Run the following commands to set the METIS environment variables:

```bash
export PATH=/path/to/METIS/bin:$PATH
export LD_LIBRARY_PATH=/path/to/METIS/lib:$LD_LIBRARY_PATH
```

---End

### 2.6.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the Code_Saturne installation package v6.0.2.tar.gz.


**Step 2** Use SFTP to upload the Code_Saturne installation package to the /path/to/Code_Saturne directory on the server.

---End

### 2.6.6 Compiling and Installing Code_Saturne

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the Code_Saturne installation package:

```bash
tar -xvf v6.0.2.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:

```bash
cd code_saturne-6.0.2
```

**Step 4** Run the following command to start bootstrap:

```bash
./sbin/bootstrap
```

**Step 5** Run the following command to perform configuration:

```bash
CC=mpicc FC=mpifort CXX=mpicxx ./configure --prefix=/path/to/CODE_SATURNE --with blas-lib=/usr/lib64 --with hdf5=/path/to/HDF5 --with scotch=/path/to/SCOTCH --with metis=/path/to/METIS --with med=/path/to/MED --enable-shared --with mpi
```

**Step 6** Run the following commands to perform compilation and installation:
make -j
make install

**Step 7** Run the following command to set the Code_Saturne environment variable:

```
export PATH=/path/to/CODE_SATURNE/bin:$PATH
```

**Step 8** Run the following command to check whether Code_Saturne is installed successfully:

```
which code_saturne
```

If information similar to the following is displayed, the installation is successful:

```
/path/to/CODE_SATURNE/bin/code_saturne
```

---End

### 2.6.7 Running and Verifying Code_Saturne

#### 2.6.7.1 Running and Verifying Code_Saturne on a Single Node

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Create a case directory and go to the directory.

```
cd /path/to/CODE_SATURNE/
mkdir TESTCASE
cd TESTCASE
```

**Step 3** Run the following command to create a case:

```
code_saturne create -s test -c case1
```

**NOTE**

In the previous command, test indicates the user-defined study name and case1 indicates the user-defined case name. You can change them based on the actual situation. After this command is run, a test directory is generated in the current directory, and the case1, MESH, and POST directories are generated in the test directory.

**Step 4** Run the following command to copy the test case file:

```
cp /path/to/CODE_SATURNE/code_saturne-6.0.2/examples/1-simple_junction/mesh/* test/MESH -rf
```

```
cp /path/to/CODE_SATURNE/code_saturne-6.0.2/examples/1-simple_junction/case1/setup.xml test/case1/DATA -rf
```

**Step 5** Run the following command to set the MPI environment variable:

```
export CS_MPIEXEC_OPTIONS="--allow-run-as-root --mca btl ^openib"
```

**Step 6** Run the following command to run Code_Saturne on a single node:

```
code_saturne run -n 18 --nt=1 --param=/path/to/CODE_SATURNE/TESTCASE/test/case1/DATA/setup.xml --case=/path/to/CODE_SATURNE/TESTCASE/test/case1
```
Table 2-30 Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n &lt;nprocs&gt;</td>
<td>Number of MPI processes.</td>
</tr>
<tr>
<td>--nt=NTHREADS</td>
<td>Number of OpenMP threads of each task.</td>
</tr>
<tr>
<td>--param=&lt;param&gt;</td>
<td>Path and name of the parameter file.</td>
</tr>
<tr>
<td>--case=&lt;case&gt;</td>
<td>Case path.</td>
</tr>
</tbody>
</table>

If the following information is displayed, the initialization is successful.

```
Post-calculation operations
```

**Step 7** Run the following command to go to the directory of test cases:

```
cd /path/to/CODE_SATURNE/TESTCASE/test/case1/RESU/year+month+day-hour+minute_round
```

--- **NOTE**

A log is generated in the `RESU` directory each time a job is run. The log directory format is `YearMonthDay-HourMinute.Sequence number of job executions in the same minute`. For example, the `20200330-2104` directory contains the log of the first job execution at 21:04 on March 30, 2020, and the `20200330-2104_1` directory contains the log of the second job execution at 21:04 on March 30, 2020.

**Step 8** View the execution result.

**Elapsed time**: `XX s` in the `performance.log` file indicates the performance. A smaller value indicates a better performance.

The following figure shows the command output.

**Figure 2-4** An example result

----End

2.6.7.2 Running and Verifying Code_Saturne on Multiple Nodes

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Create a case directory and go to the directory.
cd /path/to/CODE_SATURNE/

mkdir TESTCASE

cd TESTCASE

Step 3  Run the following command to create a case:

code_saturne create -s test -c case1

NOTE

In the previous command, test indicates the user-defined study name and case1 indicates the user-defined case name. You can change them based on the actual situation. After this command is run, a test directory is generated in the current directory, and the case1, MESH, and POST directories are generated in the test directory.

Step 4  Run the following command to copy the test case file:

cp /path/to/CODE_SATURNE/code_saturne-6.0.2/examples/1-simple_junction/mesh/* test/MESH -rf


cp /path/to/CODE_SATURNE/code_saturne-6.0.2/examples/1-simple_junction/case1/setup.xml test/case1/DATA -rf

Step 5  Run the following command to set the MPI environment variable:

export CS_MPIEXEC_OPTIONS="--allow-run-as-root --mca btl ^openib"

Step 6  Run the following command to initialize the case:

code_saturne run --initialize --param=/path/to/CODE_SATURNE/TESTCASE/test/case1/DATA/setup.xml --case=/path/to/CODE_SATURNE/TESTCASE/test/case1

If the following information is displayed, the initialization is successful.

Preprocessing calculation

Step 7  Run the following command to create the hostfile file and add node information:

echo -e 'node1\nnode2\n...\nnodex' > /path/to/HOSTFILE

NOTE

In the previous command, node1, node2, ..., and nodex indicate the host names of the nodes. You can run the hostname command to query the host names. \n indicates a newline character.

Step 8  Run the following command to go to the directory of test cases:

cd /path/to/CODE_SATURNE/TESTCASE/test/case1/RESU/year+month+day-hour+minute_round

NOTE

A log is generated in the RESU directory each time a job is run. The log directory format is YearMonthDay-HourMinute_Sequence number of job executions in the same minute. For example, the 20200330-2104 directory contains the log of the first job execution at 21:04 on March 30, 2020, and the 20200330-2104_1 directory contains the log of the second job execution at 21:04 on March 30, 2020.
Step 9  Run the following command to run the test case on multiple nodes:

```
 time mpirun --allow-run-as-root --mca btl ^openib -N 96 -x OMP_NUM_THREADS=1 --hostfile /path/to/HOSTFILE ./cs_solver --mpi
```

Step 10  View the execution result.

**Elapsed time:** XX s in the `performance.log` file indicates the performance. A smaller value indicates a better performance.

The following figure shows the command output.

![Command Output](image)

**Figure 2-5** An example result

----End

2.6.8 More Information

Official Code_Saturne website:

[https://www.code-saturne.org/cms/documentation](https://www.code-saturne.org/cms/documentation)

2.7 FDS 6.7.4 Porting Guide (CentOS 7.6)

2.7.1 Introduction

FDS, developed by the National Institute of Standards and Technology (NIST) of the U.S. Department of Commerce and the VTT Technical Research Center of Finland, is a large eddy simulation software for low-speed flows. It focuses on the smoke and heat transfer caused by fires and aims to solve the actual fire problems in fire engineering. It also provides tools for studying basic dynamics and combustion in a fire.

For more information about FDS, visit the [official FDS website](https://www.code-saturne.org/cms/documentation).

Programming language: Fortran

Brief description: fire smoke and heat transfer simulation tool

**Recommended Version**

FDS 6.7.4
2.7.2 Environment Requirements

Hardware Requirements

Table 2-31 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 2-32 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDS</td>
<td>6.7.4</td>
<td><a href="https://codeload.github.com/firemodels/fds/tar.gz/FDS6.7.4">https://codeload.github.com/firemodels/fds/tar.gz/FDS6.7.4</a></td>
</tr>
<tr>
<td>Test case</td>
<td>weak_scaling_test_128</td>
<td>Test case provided by the software</td>
</tr>
</tbody>
</table>

OS Requirements

Table 2-33 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

2.7.3 Planning the Paths for Software Porting

Table Paths for software porting lists the software installation paths involved in the FDS software porting.
Table 2-34 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see section Planning Data for Installation in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/FDS</td>
<td>Installation path of FDS</td>
<td>The installation path listed in this table is only an example. A shared path is recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

2.7.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 2-35 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Cluster Environment&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

2.7.5 Obtaining the Source Code

Procedure

Step 1  Download the FDS installation package fds-FDS6.7.4.tar.gz.

URL: [https://codeload.github.com/firemodels/fds/tar.gz/FDS6.7.4](https://codeload.github.com/firemodels/fds/tar.gz/FDS6.7.4).

Step 2  Use an SFTP tool to upload the FDS installation package to the /path/to/FDS directory on the server.

----End
2.7.6 Compiling and Installing FDS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to load environment variables:

```
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Run the following command to decompress the FDS installation package:

```
tar zxvf fds-FDS6.7.4.tar.gz
```

Step 4 Run the following command to switch to the directory generated after decompression:

```
cd fds-FDS6.7.4
```

Step 5 Run the following commands to set the `-m64 -O2` compilation option in the `makefile` file to `-mabi=lp64 -march=armv8.2-a -mtune=tsv110 -O3 -flto`:

```
cd /path/to/FDS/fds-FDS6.7.4/Build
sed -i "s/-m64 -O2/-mabi=lp64 -march=armv8.2-a -mtune=tsv110 -O3 -flto/g" ./makefile
```

Step 6 Run the following command to switch to the FDS compilation directory:

```
cd /path/to/FDS/fds-FDS6.7.4/Build/mpi_gnu_linux_64
```

Step 7 Run the following command to run the automatic compilation script:

```
./make_fds.sh
```

Step 8 Run the following commands to add the FDS environment variable:

```
export PATH=/path/to/FDS/fds-FDS6.7.4/Build/mpi_gnu_linux_64:$PATH
```

----End

2.7.7 Running and Verifying FDS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Go to the FDS root directory.

```
cd /path/to/FDS/fds-FDS6.7.4/
```
Step 3 Run the following command to create the **testcase** directory:
```
mkdir testcase
```

Step 4 Run the following command to go to the **testcase** directory:
```
cd testcase
```

Step 5 Run the following command to set the number of OpenMP threads to 1:
```
export OMP_NUM_THREADS=1
```

Step 6 Run the following command to create the **hostfile** file and add node information:
```
echo -e 'node1\nnode2\n...\nnodex' > /path/to/HOSTFILE
```

**NOTE**
- In the preceding command, `node1`, `node2`, ..., and `nodex` indicate the host names of the nodes. You can run the `hostname` command to query the host names. \n indicates a newline character.
- This step is mandatory when multiple nodes are running and can be skipped when only one node is running.

Step 7 Run the following commands to run FDS:
- **Single-node:**
  ```
  `which mpirun` -np 128 --mca btl ^openib --allow-run-as-root ../Build/
  MPI_Scaling_Tests/FDS_Input_Files/weak_scaling_test_128.fds
  ```
- **Multi-node:**
  ```
  `which mpirun` -N 128 -x OMP_NUM_THREADS -x PATH -x
  LD_LIBRARY_PATH --hostfile /path/to/HOSTFILE --mca btl ^openib --
  allow-run-as-root ../Build/mpi_gnu_linux_64/fds_mpi_gnu_linux_64 ..//
  Validation/MPI_Scaling_Tests/FDS_Input_Files/weak_scaling_test_288.fds
  ```

**NOTE**
In the preceding case, the value of `X` in `weak_scaling_test_X.fds` must be greater than the total number of CPU cores used for the running. For example, if two nodes need to use test cases whose `X` is greater than or equal to 256, set `weak_scaling_test_X.fds` to `weak_scaling_test_288.fds`.

If the command is successfully run on a single node, the following information is displayed:
```
STOP: FDS completed successfully (CHID: weak_scaling_test_128)
```

Check the value of **real** in the **fds.log** log. The unit is second. A smaller value indicates better performance.

**Figure 2-6** shows an example.
2.7.8 Troubleshooting

Failure 1: FDS Runtime Error

Symptom
An error is reported during FDS running, indicating that too many files are opened. The error message is "Fortran runtime error: Cannot open file 'weak_scaling_test_128_0022_04.s3d': Too many open files."

Possible Causes
When the FDS is running, a large number of files are read and written. By default, the maximum number of files that can be opened is 1024, which cannot meet the FDS running requirements.

Procedure
Tens of millions of files can be opened at the same time on a Taishan server. You can run the `ulimit -n` command to set the maximum number of files that can be opened at the same time.

For example, run the `ulimit -n 10240` command.

2.7.9 More Information

FDS home page:
https://pages.nist.gov/fds-smv/

FDS usage guidelines:

2.8 Elmer 8.4 Porting Guide (CentOS 7.6)

2.8.1 Introduction

Elmer is a finite element software for numerical solutions of partial differential equations. Elmer can handle any number of equations, so it is very suitable for simulating multiphysical problems. Elmer includes physical models of fluid dynamics, structural mechanics, electromagnetics, heat transfer and acoustics.
Users can also write their own equations that can be dynamically linked to the main program.

For more information about Elmer, visit the official Elmer website.

Language: Fortran, C++, C

Brief description: multiphysical simulation software

Open-source license: GPLv2

**Recommended Version**

Elmer 8.4

### 2.8.2 Environment Requirements

#### Hardware Requirements

*Table 2-36* lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

#### Software Requirements

*Table 2-37* lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elmer</td>
<td>8.4</td>
<td><a href="https://codeload.github.com/ElmerCSC/elmerfem/tar.gz/scc20">https://codeload.github.com/ElmerCSC/elmerfem/tar.gz/scc20</a></td>
</tr>
<tr>
<td>CMake</td>
<td>3.8.1</td>
<td><a href="https://cmake.org/download/">https://cmake.org/download/</a></td>
</tr>
<tr>
<td>OpenBLAS</td>
<td>0.3.9</td>
<td><a href="https://github.com/xianyi/OpenBLAS/archive/v0.3.9.tar.gz">https://github.com/xianyi/OpenBLAS/archive/v0.3.9.tar.gz</a></td>
</tr>
</tbody>
</table>

#### OS Requirements

*Table 2-38* lists the OS requirements.
### Table 2-38 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 2.8.3 Planning the Paths for Software Porting

This section describes the software installation paths involved in the Elmer software porting.

#### Table 2-39 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/ELMER</td>
<td>Installation path of Elmer.</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/CMAKE</td>
<td>Installation path of Cmake.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS.</td>
<td></td>
</tr>
</tbody>
</table>

### 2.8.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.
Configuration Process

Table 2-40 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing CMake.</td>
<td>For details, see 2.8.4.1 Installing CMake.</td>
</tr>
<tr>
<td>3</td>
<td>Installing OpenBLAS.</td>
<td>For details, see 2.8.4.2 Installing OpenBLAS.</td>
</tr>
</tbody>
</table>

2.8.4.1 Installing CMake

Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to decompress the CMake installation package:

```
tar zxvf cmake-3.8.1.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:

```
cd cmake-3.8.1
```

**Step 4** Run the following command to perform configuration:

```
./configure --prefix=/path/to/CMAKE
```

**Step 5** Run the following commands to perform compilation and installation:

```
make
make install
```

**Step 6** Run the following commands to set the environment variable of CMake:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

---End

2.8.4.2 Installing OpenBLAS

Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to decompress the installation package:
Step 3 Run the following command to switch to the directory generated after decompression:
   cd OpenBLAS-0.3.9/

Step 4 Run the following commands to set environment variables:
   export CC=`which gcc`
   export CXX=`which g++`
   export FC=`which gfortran`

Step 5 Run the following commands to perform compilation and installation:
   make -j 96
   make PREFIX=/path/to/OPENBLAS install

Step 6 Run the following commands to set the environment variables of OpenBLAS:
   export LIBRARY_PATH=/path/to/OPENBLAS/lib:$LIBRARY_PATH
   export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH

2.8.5 Obtaining the Source Code

Procedure

Step 1 Download the Elmer source code package elmerfem-scc20.tar.gz.
   URL: https://codeload.github.com/ElmerCSC/elmerfem/tar.gz/scc20

Step 2 Download the official test case package ElmerTutorialsFiles_nonGUI.tar.gz
   URL: http://www.nic.funet.fi/pub/sci/physics/elmer/doc/ElmerTutorialsFiles_nonGUI.tar.gz

Step 3 Use SFTP to upload the downloaded packages to the /path/to/ELMER/ directory on the server.

2.8.6 Compiling and Installing Elmer

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the Elmer installation package:
   tar -xvf elmerfem-scc20.tar.gz

Step 3 Run the following command to switch to the directory generated after decompression:
   cd elmerfem-scc20
Step 4 Run the following commands to create a folder:

```bash
mkdir build
cd build
```

Step 5 Run the following command to perform compilation:

```bash
cmake -DCMAKE_INSTALL_PREFIX=/path/to/ELMER -DCMAKE_C_FLAGS="-O3 -march=armv8.2-a -mtune=tsv110" -DCMAKE_CXX_FLAGS="-O3 -march=armv8.2-a -mtune=tsv110" -DCMAKE_Fortran_FLAGS="-O3 -march=armv8.2-a -mtune=tsv110" ../
```

Step 6 Run the following commands to perform installation:

```bash
make -j 32
make install
```

Step 7 Run the following commands to set environment variables:

```bash
export PATH=/path/to/ELMER/bin:$PATH
export ELMER_HOME=/path/to/ELMER
```

---End

### 2.8.7 Running and Verifying Elmer

#### Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to create the hostfile file and add node information:

```bash
echo -e 'node1
node2
…
nodex' > /path/to/HOSTFILE
```

**NOTE**
- In the preceding command, node1, node2, …, and nodex indicate the host names of the nodes. You can run the `hostname` command to query the host names. `\n` indicates a newline character.
- This step is mandatory when multiple nodes are running and can be skipped when only one node is running.

**Step 3** Run the following commands to decompress the official package:

```bash
cd /path/to/ELMER/
tar -xvf ElmerTutorialsFiles_nonGUI.tar.gz
```

**Step 4** Run the following command to enter the test case:

```bash
cd tutorials_files/FlowResistance
```

**NOTE**
- FlowResistance is one of the official examples.
- For details about how to run other test cases, see the official manual.
Step 5  Run the following command to generate a grid:

\texttt{ElmerGrid 1 2 hole.grd}

Step 6  Run the following command to run the case:

\texttt{time mpirun --allow-run-as-root -mca btl ^openib --hostfile /path/to/HOSTFILE -x PATH -x LD_LIBRARY_PATH -x ELMER_HOME -N 128 ElmerSolver}

\textbf{NOTE}

Run the following command to run the case in single-node mode:

\texttt{time mpirun --allow-run-as-root -mca btl ^openib -np 128 ElmerSolver}

After the command is run, the following information is displayed:

\begin{tabular}{l}
\texttt{ElmerSolver: *** Elmer Solver: ALL DONE ***} \\
\texttt{ElmerSolver: The end} \\
\texttt{SOLVER TOTAL TIME(CPU,REAL): 10.23 2.86} \\
\texttt{ELMER SOLVER FINISHED AT: 2020/05/12 17:16:54}
\end{tabular}

---End

2.8.8 Troubleshooting

Problem 1: An Error Is Reported When make install Is Run

\textbf{Symptom}

"Error: Array specification at (1) has more than 7 dimensions" is displayed when the \texttt{make install} command is run.

\textbf{Possible Cause}

\begin{itemize}
\item The environment configuration is incorrect during compilation.
\end{itemize}

\textbf{Procedure}

After consultation and internet searches, the error is caused by the gfortran version that is too early. Add the following command to recompile the environment variable:

\texttt{export FC= /path/to/GNU/bin/gfortran}

2.8.9 More Information

For more information, visit the official Elmer website:

\texttt{http://www.csc.fi/elmer}

2.9 Code_Aster 14.6.0 Porting Guide (CentOS 7.6)

2.9.1 Introduction

Code_Aster is a software package for finite element analysis and numeric simulation in structural mechanics. It runs on Linux and was public released based on General Public License in October 2001. Code_Aster has now 1,500,000 lines of source code, most of which are developed in Fortran and Python, and is being
constantly developed and updated. Most of the Code_Aster fields have been validated by independent comparisons with analytical or experimental results, benchmarks towards other codes.

Code_Aster is a solver, based on the theory of the mechanics of the continuous media, which uses the method of the finite elements to solve different types of mechanical, thermal, acoustic, seismic, and other problems. Beyond the standard functionalities of a FEM software for solid mechanics, Code_Aster also compiles specific research in various fields: fatigue, damage, fracture, contact, geomaterials, porous media, and multiphysics coupling.

Currently, the Code_Aster software has been widely used to evaluate and maintain power plants and power networks, and to analyze and calculate bridge data.

Salome is an open-source software that provides a generic platform for pre- and post-processing for numerical simulation. Salome-Meca is a standalone application, which is an external project based on the SALOME platform. Salome-Meca represents integration of the Code_Aster solver in the SALOME platform.

Programming languages: C/Fortran/Python

Brief description: Code_Aster is a software package for finite element analysis and numeric simulation in structural mechanics.

Open source license: GPL

Recommended Software Version

Code_Aster 14.6.0

2.9.2 Environment Requirements

Hardware Requirements

Table 2-41 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 2-42 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
</table>
OS Requirements

Table 2-43 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

2.9.3 Paths for Software Porting

This chapter describes the software installation paths involved in the Code_Aster software porting.

Table 2-44 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/path/to/KPGCC</td>
<td>Installation path of the Kunpeng GCC.</td>
<td>For details, see section &quot;Kunpeng GCC Installation and Operation&quot; in User Guide (Kunpeng GCC).</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HMPI</td>
<td>Installation path of the Hyper MPI.</td>
<td>For details, see section &quot;Source Code Installation&quot; in Hyper MPI User Guide.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/CODE-ASTER</td>
<td>Installation path of Code_Aster.</td>
<td>The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

2.9.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.
Configure Process

Table 2-45 Configure process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the open source environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>Setting up the Huawei Kunpeng compiler environment</td>
<td>For details, see section &quot;Installing the Kunpeng GCC&quot; in <a href="#">User Guide (Kunpeng GCC)</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Setting up the Huawei MPI environment</td>
<td>For details, see section &quot;Installation Introduction&quot; in <a href="#">Hyper MPI User Guide</a>.</td>
</tr>
</tbody>
</table>

2.9.4.1 Configuring the Yum Source

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to download the Base source configuration file:

```
wget --no-check-certificate -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.tools.huawei.com/repository/conf/CentOS-AltArch-7.repo
```

**Step 3** Run the following commands to configure the EPEL source configuration file:

```
echo '
name=Extra Packages for Enterprise Linux 7 - $basearch
baseurl=http://mirrors.tools.huawei.com/rep/7/$basearch
failovermethod=priority
enabled=1
gpgcheck=0' > /etc/yum.repos.d/epel.repo
```

**Step 4** Run the following commands to refresh the Yum source cache:

```
yum clean all
yum makecache
----End
```
2.9.5 Obtaining the Source Code

Procedure

**Step 1** Download the Code_Aster installation package `aster-full-src-14.6.0-1.noarch.tar.gz`.
Download address: [https://www.code-aster.org/FICHIERS/aster-full-src-14.6.0-1.noarch.tar.gz](https://www.code-aster.org/FICHIERS/aster-full-src-14.6.0-1.noarch.tar.gz)

**Step 2** Use an SFTP tool to upload the Code_Aster installation package to the `/path/to/CODE-ASTER` directory on the server.

----End

2.9.6 Compilation and Installation

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to install the dependencies:

```
yum -y install zlib* lapack* blas* python3.aarch64 python3-devel.aarch64 cmake3* boost_python* boost boost-devel numpy* python36-numpy flex bison
```

**Step 3** Run the following command to go to the main program installation directory:

```
cd /path/to/COPE-ASTER
```

**Step 4** Run the following command to decompress the main program installation package:

```
tar -xvf aster-full-src-14.6.0-1.noarch.tar.gz
```

**Step 5** Run the following command to go to the directory generated after decompression:

```
 cd aster-full-src-14.6.0
```

**Step 6** Run the following commands to modify the `setup.py` file:

1. Open the `setup.py` file.

   ```
   vi setup.py
   ```

2. Press `i` to go to the edit mode and add the `aarch64` option.
3. Press **Esc**, type `:wq!`, and press **Enter** to save the file and exit.

**Step 7** Run the following commands to modify the `as_setup.py` file:

1. Open the `as_setup.py` file.
   
   **vi as_setup.py**

2. Press i to go to the edit mode.

   a. Add the `aarch64` option to line 1538.

   ```python
   'aarch64': 'shell|script|aarch64',
   ```

   b. Modify the number of compilation cores to increase the compilation speed.
3. Press Esc, type :wq!, and press Enter to save the file and exit.

**Step 8**  Run the following commands to modify the setup.cfg file:

1. Open the setup.cfg file.
   
   `vi setup.cfg`

2. Press i to go to the edit mode and modify the file as follows:
   
   ```
   CC='/path/to/gcc/bin/gcc'
   CXX='/path/to/gcc/bin/g++'
   F90='/path/to/gcc/bin/gfortran'
   LD=F90
   CXXFLAGS="-std=c++11"
   CXXLIB='-L/path/to/gcc/lib64 -lstdc++'
   ```

3. Press Esc, type :wq!, and press Enter to save the file and exit.

**Step 9**  Run the following commands to modify the med configuration:

```bash
sed '186 a\ conf_cmd = ("export F77=$F90; ";' -i products.py
sed -i '188d' products.py
```

**Step 10**  Run the following command to go to the Code_Aster source code directory:

`cd SRC/`

**Step 11**  Run the following command to decompress the Code_Aster installation package:

```bash
tar xvf aster-14.6.0.tgz
```
Step 12  Run the following commands to edit the files:

1. Open the /bibfor/wscript file.
   
   vi aster-14.6.0/bibfor/wscript

2. Press i to go to the edit mode and modify line 103 as follows:
   
   if self.env.DEST_CPU == 'x86_64' or self.env.DEST_CPU == 'aarch64':

3. Press Esc, type :wq!, and press Enter to save the file and exit.

   
   vi /aster-14.6.0/bibpyt/Utilitai/Utmess.py

5. Press i to go to the edit mode and modify line 224 as follows:
   
   curr_idmess = idmess.split("\0")
   x = curr_idmess[0].split("_")
   # x = idmess.split("_")

6. Press Esc, type :wq!, and press Enter to save the file and exit.

Step 13  Run the following command to decompress the Code_Aster installation package:

   tar cvzf aster-14.6.0.tgz aster-14.6.0

Step 14  Run the following command to delete the Code_Aster installation package:

   rm -rf aster-14.6.0

Step 15  Run the following command to perform the compilation:

   cd /path/to/CODE-ASTER/aster-full-src-14.6.0
   echo y | python3 setup.py --prefix=/path/to/CODE-ASTER
2.9.7 Running and Verification

Procedure

Step 1 Run the following command to go to the CODE-ASTER directory:

```
cd /path/to/CODE-ASTER
```

Step 2 Run the following command to configure the environment variables:

```
source /path/to/CODE-ASTER/profile.sh
```

Step 3 Run the following command to obtain the forma01a.export file:

```
as_run --get_export forma01a > forma01a.export
```

Step 4 Run the following commands to modify the forma01a.export file:

1. Open the forma01a.export file.

```
vi forma01a.export
```

2. Press `i` to go to the edit mode and change the value of `memory_limit` in line 5 to `2048.0`.

3. Press Esc, type `:wq!`, and press Enter to save the file and exit.
Step 5  Run the following command to execute the configuration file:

```bash
as_run forma01a.export
```

![Command output](image)

----End

2.9.8 More Resources

Code_Aster official website:

[https://www.code-aster.org](https://www.code-aster.org)
3 Government HPC

3.1 udunits 2.2.28 Porting Guide (CentOS 7.6)
3.2 CANU 1.8 Porting Guide (CentOS 7.6)
3.3 GROMACS 2019.3 Porting Guide (CentOS 7.6)
3.4 LAMMPS 5 Jun 2019 Porting Guide (CentOS 7.6)
3.5 Quantum ESPRESSO 6.4.1 Porting Guide (CentOS 7.6)
3.6 NAMD 2.13 Porting Guide (CentOS 7.6)
3.7 VASP 5.4.4 Porting Guide (CentOS 7.6)
3.8 ABINIT 8.10.3 Porting Guide (CentOS 7.6)
3.9 AmberTools 19 Porting Guide (CentOS 7.6)
3.10 CP2K 4.1 Porting Guide (CentOS 7.6)
3.11 CP2K 7.1 Porting Guide (CentOS 7.6)
3.12 NWChem 6.8.1 Porting Guide (CentOS 7.6)
3.13 ROOT 6.20 Porting Guide (CentOS 7.6)
3.14 Geant4 10.6 Porting Guide (CentOS 7.6)
3.15 Ont-Tombo 1.5.1 Porting Guide (CentOS 7.6)
3.16 DL_POLY 1.10 Porting Guide (CentOS 7.6)
3.17 Gamess Porting Guide (CentOS 7.6)
3.18 CPMD 4.1 Porting Guide (CentOS 7.6)
3.19 ESPResSo 4.1.4 Porting Guide (CentOS 7.6)
3.20 MOOSE Framework 1.0.0 Porting Guide (CentOS 7.6)
3.21 MPB 1.11.1 and Meep 1.17.1 Porting Guide (CentOS 7.6)
3.22 Octopus 10.3 Porting Guide (CentOS 7.6)
3.23 Psi4 1.3.2 Porting Guide (CentOS 7.6)
3.1 udunits 2.2.28 Porting Guide (CentOS 7.6)

3.1.1 Introduction

The UDUNITS-2 package provides support for units of physical quantities. Its three main components are: 1) a C library for units of physical quantities (udunits2lib); 2) a utility, for obtaining the definition of a unit and for converting numeric values between compatible units (udunits2prog), and 3) an extensive database of units.

For more information about UDUNITS-2, visit the [official UDUNITS-2 website](https://www udunits.org).

Programming language: C/Fortran

One-sentence description: databases

Open-source protocol: GPLv3

Recommended Version

The recommended version is udunits-2.2.28.

3.1.2 Environment Requirements

Hardware Requirements

Table 3-1 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-2 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
</table>
### OS Requirements

Table 3-3 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 3.1.3 Planning Paths for Software Porting

This section describes the software installation paths involved in the UDUNITS software porting.

Table 3-4 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/UDUNITS</td>
<td>Installation path of UDUNITS.</td>
<td>The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>
3.1.4 Configuring the Compilation Environment

Prerequisites

Installation packages are uploaded to a server using an SFTP tool.

Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Basic environment setup</td>
<td>For details, see &quot;Environment Setup in Cluster Scenarios&quot; in the <em>HPC Solution Basic Environment Setup Guide.</em></td>
</tr>
</tbody>
</table>

3.1.5 Obtaining the Source Code

Procedure

Step 1 Obtain UDUNITS installation package `udunits-2.2.28.tar.gz` from the following address:

https://artifacts.unidata.ucar.edu/repository/downloads-udunits/udunits-2.2.28.tar.gz

Step 2 Use the SFTP tool to upload the UDUNITS installation package to the `/path/to/UDUNITS` directory on the server.

----End

3.1.6 Compiling and Installing UDUNITS

Procedure

Step 1 Use PuTTY to log in to the server as the *root* user.

Step 2 Run the following command to go to the main program installation directory:

```
    cd /path/to/UDUNITS
```

Step 3 Run the following command to decompress the package:

```
    tar -zxvf udunits-2.2.28.tar.gz
```

Step 4 Run the following command to switch to the directory generated after decompression:

```
    cd udunits-2.2.28
```

Step 5 Run the following command to perform configuration:

```
    ./configure --prefix=/path/to/UDUNITS CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```
3.1.7 Running and Verifying UDUNITs

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check the installation path. If the following file is generated, the installation is successful:

```bash
#!/bin/
udunits2
#include;
converter.h udunits2.h udunits.h
#lib;
libudunits2.a libudunits2_la libudunits2.so libudunits2.so.0 libudunits2.so.0.1.0
#share;
info info udunits
```

3.2 CANU 1.8 Porting Guide (CentOS 7.6)

3.2.1 Introduction

CANU is a third-generation data assembly tool designed by Celera Assembler. It supports PacBio RSII and Oxford Nanopore MinION and is used for high-noise single-molecule sequencing. The main functions include correction, trimming, and stitching.

For more information about CANU, visit the official CANU website.

Language: C/Perl

Brief description: high-noise single-molecule sequencing tool

Open-source protocol: GPL 2.0

Recommended Version

The recommended version is CANU 1.8.

3.2.2 Environment Requirements

Hardware Requirements

Table 3-6 lists the hardware requirements.
Table 3-6 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-7 lists the requirements on the OS.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Contained in the OS image.</td>
</tr>
</tbody>
</table>

3.2.3 Planning the Paths for Software Porting

The following table lists the software installation paths involved in the CANU 1.8 software porting.

Table 3-8 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/path/to/GNU</td>
<td>Installation path of GNU 9.1</td>
<td>The planned installation paths are only examples. You are advised to deploy the software in shared paths. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/CANU</td>
<td>Installation path of CANU 1.8</td>
<td></td>
</tr>
</tbody>
</table>

3.2.4 Configuring the Compilation Environment

Step 1 Install the GNU and configure environment variables.

For details about how to install the GNU 9.1 compiler, see the GNU 9.1 Installation Guide.

Step 2 Use PuTTY to log in to a server as the root user.

Step 3 Run the following commands to load the environment variables of GNU 9.1:
Step 4 Run the following command to install Boost:

```bash
yum install boost* -y
```

NOTE

The YUM source is a local source of the CentOS 7.6 image. You can change the files in the `/etc/yum.repos.d` directory as required.

---End

### 3.2.5 Obtaining the Source Code

**Step 1** Download the CANU source code package `canu-1.8.tar.gz`.

Download address: [https://github.com/marbl/canu/releases](https://github.com/marbl/canu/releases)

**Step 2** Use the SFTP tool to upload the CANU source code package to the `/path/to/CANU` directory on the server.

NOTE

For details about `/path/to/CANU`, see Table 3-8.

---End

### 3.2.6 Compiling and Installing CANU

**Step 1** Run the following commands on PuTTY to decompress the CANU installation package:

```
tar -xvf canu-1.8.tar.gz
```

**Step 2** Run the following command to switch to CANU source code directory:

```
cd canu-1.8/src
```

**Step 3** Run the following command to compile CANU:

```
make -j8
```

**Step 4** Go to the CANU 1.8 directory. If the CANU executable file is generated in the `Linux-aarch64/bin` directory, as shown in the following command output, the compilation is complete.

```
-rwxrwxr-x 1 root root 31114 Oct 23  2018 canu
```

**Step 5** Run the following command to add CANU to the environment variable:

```
export PATH=/path/to/CANU/canu-1.8/Linux-aarch64/bin:$PATH
```
3.2.7 Running and Verifying CANU

Step 1 Run the following command on PuTTY to install Java 1.8.0:

```bash
yum install java-1.8.0-openjdk* gnuplot -y
```

**NOTE**
The YUM source is a local source of the CentOS 7.6 image. You can change the files in the `/etc/yum.repos.d` directory as required.

Step 2 Run the following command to obtain the case file:

```bash
curl -L -o oxford.fasta http://gembox.cbcb.umd.edu/mhap/raw/ecoli_p6_25x.filtered.fastq
```

**NOTE**
Access to the external network is required. You can download and convert the case file by using a server that is connected to the external network, and then copy the case file to the test directory.

Step 3 Run the following command to start CANU:

```bash
{ time canu -p ecoli -d ecoli-pacbio genomeSize=4.8m corPartitions=384 corMemory=3 corPartitionMin=1000 corThreads=1 useGrid=false obtovlThreads=96 obtOvlHashBlockLength=682496000 utgOvlHashBlockLength=682496000 utgovlThreads=96 gridEngineMemoryOption="-l vf=MEMORY" -nanopore-raw oxford.fastq ; } 2>&1 |tee -a canu.log
```

**Table 3-9** describes the command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>useGrid</td>
<td>Whether to run under Grid Control. The default value is <strong>true</strong>. Change the value to <strong>false</strong>.</td>
</tr>
<tr>
<td>-p assembly-prefix</td>
<td>assembly-prefix component index. Set this parameter based on the case.</td>
</tr>
<tr>
<td>-d assembly-directory</td>
<td>assembly-directory component directory. The value of this parameter is user-defined.</td>
</tr>
<tr>
<td>genomeSize</td>
<td>Genome size. Set this parameter based on the case.</td>
</tr>
<tr>
<td>-nanopore-raw</td>
<td>Select the required gene file format based on the case.</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>corPartitions</td>
<td>Number of tasks. You are advised to set this parameter to a multiple of the number of cores. The recommended value is 4 times the number of cores.</td>
</tr>
<tr>
<td>corThreads</td>
<td>Number of threads in each process. The recommended value is 1.</td>
</tr>
<tr>
<td>obtovlThreads</td>
<td>Number of threads for running obtovl. You are advised to set this parameter to the number of cores.</td>
</tr>
<tr>
<td>utgovlThreads</td>
<td>Number of threads for running utgov. You are advised to set this parameter to the number of cores.</td>
</tr>
</tbody>
</table>

---

### 3.3 GROMACS 2019.3 Porting Guide (CentOS 7.6)

#### 3.3.1 Introduction

Molecular dynamics (MD) simulation is the most widely used method for calculating a large complex system. Due to rapid development of molecular simulation since 1970, many force fields applicable to biochemical molecular systems, polymers, metals, and non-metallic materials had been systematically established. This greatly improves the capability and accuracy of computing complex system structures and some thermodynamic and spectral properties. MD simulation is a computational method developed by applying these force fields and according to Newton's mechanics of motion.

GROningen MAchine for Chemical Simulations (GROMACS) is a computational engine for MD simulation and energy minimization, which simulates hundreds to millions of atomic systems using Newton's equilibrium equations. It is primarily designed for biochemical molecules like proteins, lipids and nucleic acids that have a large number of complicated bonded interactions, but since it is extremely fast at calculating the nonbonded interactions (that usually dominate simulations), many groups are also using it for research on non-biological systems, such as polymers. Compared with other MD simulation software, GROMACS has the following unique advantages:

1. GROMACS is free of charge and complies with the GNU Lesser General Public License (LGPL) protocol. You can find the open-source code of GROMACS on GitHub.
2. GROMACS provides higher performance than other software and has been optimized in code.
3. GROMACS is user-friendly for topology files and parameter setting files. The format of GROMACS is similar to that of Python.
4. The environment of GROMACS develops well, and many simulation tools support GROMACS well.

For more information about GROMACS, visit the official GROMACS website.
Recommended Version

The recommended version is GROMACS 2019.3.

3.3.2 Environment Requirements

Hardware Requirements

Table 3-10 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-11 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenBLAS</td>
<td>0.3.6</td>
<td><a href="https://github.com/xianyi/OpenBLAS/releases">https://github.com/xianyi/OpenBLAS/releases</a></td>
</tr>
<tr>
<td>FFTW</td>
<td>3.3.8</td>
<td><a href="http://www.fftw.org/download.html">http://www.fftw.org/download.html</a></td>
</tr>
<tr>
<td>cmake</td>
<td>3.8.1</td>
<td><a href="https://cmake.org/download/#previous">https://cmake.org/download/#previous</a></td>
</tr>
<tr>
<td>Test case</td>
<td>water_GMX50_bare</td>
<td><a href="https://ftp.gromacs.org/pub/benchmarks/">https://ftp.gromacs.org/pub/benchmarks/</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-12 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
</tbody>
</table>
### 3.3.3 Planning the Paths for Software Porting

Table 3-13 lists the software installation paths involved in the GROMACS software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/FFTW</td>
<td>Installation path of FFTW</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/GROMACS</td>
<td>Installation path of GROMACS</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/CASE</td>
<td>Planned path for storing the algorithm cases of the GROMACS</td>
<td></td>
</tr>
</tbody>
</table>

### 3.3.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.
**Procedure**

Table 3-14 Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Install OpenBLAS.</td>
<td>For details, see 3.3.4.1 Installing OpenBLAS.</td>
</tr>
<tr>
<td>3</td>
<td>Install FFTW.</td>
<td>For details, see 3.3.4.2 Installing FFTW.</td>
</tr>
<tr>
<td>4</td>
<td>Install CMake.</td>
<td>For details, see 3.3.4.3 Installing CMake.</td>
</tr>
</tbody>
</table>

### 3.3.4.1 Installing OpenBLAS

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the OpenBLAS installation package:

```bash
tar -xvf OpenBLAS-0.3.6.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd OpenBLAS-0.3.6
```

**Step 4** Run the following commands to set environment variables:

```bash
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

**Step 5** Run the following commands to compile and install OpenBLAS:

```bash
make
make PREFIX=/path/to/OPENBLAS install
```

**Step 6** Run the following commands to set the OpenBLAS environment variables:

```bash
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:SLD_LIBRARY_PATH
```

----End
3.3.4.2 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the FFTW installation package:

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd fftw-3.3.8
```

Step 4 Run the following command to run the bootstrap.sh script:

```
./bootstrap.sh
```

Step 5 Run the following command to perform configuration:

```
./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon --enable-float
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j40
make install
```

Step 7 Run the following commands to set FFTW environment variables:

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

----End

3.3.4.3 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the CMake installation package of version 3.8.1:

```
tar zxvf cmake-3.8.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd cmake-3.8.1
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/CMAKE
```

NOTE

The version of CMake must be 3.4.3 or later for GROMACS 2019.3.
Step 5 Run the following commands to perform compilation and installation:

   make
   make install

Step 6 Run the following commands to set the CMake environment variables:

   export PATH=/path/to/CMAKE/bin:$PATH

----End

3.3.5 Obtaining the Source Code

Procedure

Step 1 Download the GROMACS installation package gromacs-2019.3.tar.gz.

Step 2 Use SFTP to upload the GROMACS installation package to the /path/to/GROMACS directory on the server.

----End

3.3.6 Compiling and Installing GROMACS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the GROMACS installation package:

   tar -xvf gromacs-2019.3.tar.gz

Step 3 Run the following command to create the build directory:

   cd Gromacs-2019.3
   mkdir build

Step 4 Run the following command to perform configuration:

   cd build

   FLAGS="-mcpu=tsv110"; CFLAGS=$FLAGS CXXFLAGS=$FLAGS LDFLAGS="-lgfortran" CC=mpicc CXX=mpicxx \\
   /path/to/CMAKE/bin/cmake \\
   -DCMAKE_INSTALL_PREFIX=/path/to/GROMACS/gromacs-2019.3 \\
   -DBUILD_SHARED_LIBS=on \\
   -DBUILD_TESTING=on \\
   -DREGRESSIONTEST_DOWNLOAD=off \\
   -DGMX_BUILD_OWN_FFTW=off \

Kunpeng BoostKit for HPC
Porting Guide
3 Government HPC

Issue 14 (2021-07-23)   Copyright © Huawei Technologies Co., Ltd. 243
Step 5 Run the following commands to perform compilation and installation:

```
make -j40 V=1
make -j40 install
```

Step 6 Run the following command to check whether an executable file is generated:

```
ll /path/to/GROMACS/bin/gmx_mpi
```

```
-rwxr-xr-x 1 root root 83566 Jul 31 02:51 gmx_mpi
```

Step 7 Run the following commands to set environment variables:

```
export PATH=/path/to/GROMACS/bin:$PATH
```

### 3.3.7 Running and Verifying GROMACS

#### Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to switch to the directory where the case is stored:

```
cd /path/to/CASE
```

**Step 3** Decompress the compute instance files:

```
tar xvf water_GMX50_bare.tar.gz
```

**Step 4** Run the following command to switch to the directory generated after the package is decompressed:

```
cd water-cut1.0_GMX50_bare/0768
```
**Step 5** Run the following command to generate the `topol.tpr` file:

```
gmx_mpi grompp -f pme.mdp
```

**Step 6** Run the following command to view whether the `topol.tpr` file is generated:

```
ll topol.tpr
```

```
-rw-r--r-- 1 root root 1844672 Jan 11 16:41 topol.tpr
```

**Step 7** Run the following command to start the GROMACS test:

```
mpirun --allow-run-as-root --mca btl ^openib -np 96 gmx_mpi mdrun -dlb yes
-v -nsteps 10000 -resethway -noconfout -pin on -ntomp 1 -s topol.tpr
```

View the value of `ns/day` in **Performance** in the `md.log` file. The unit is **ns/day**. The higher the value is, the better the performance is.

The following is an example of the test result:

- **Part of the total run time spent waiting due to load imbalance**: 1.1%.
- **Steps where the load balancing was limited by** `-rdd`, `-rcon` and/or `-dds`: X 0 % Y 0 %
- **Average PME mesh/force load**: 1.033
- **Part of the total run time spent waiting due to PP/PME imbalance**: 2.1 %
- **Core t (s) Wall t (s) (%)**
  - Time: 14806.100 154.231 9600.0
  - (ns/day) (hour/ns)
- **Performance**: 5.603 4.283
- GROMACS reminds you: “Come on boys, Let’s push it hard” (P.J. Harvey)

----End

### 3.3.8 More Information

- Zenodo website: [https://zenodo.org/record/3243834#.XUJg-vlzZhE](https://zenodo.org/record/3243834#.XUJg-vlzZhE)

### 3.4 LAMMPS 5 Jun 2019 Porting Guide (CentOS 7.6)

#### 3.4.1 Introduction

Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) is a molecular dynamics program from Sandia National Laboratories. LAMMPS makes use of Message Passing Interface (MPI) for parallel communication and is free and open-source software, distributed under the terms of the GNU General Public License. The latest LAMMPS version supports CUDA and OpenCL-based GPU computing. LAMMPS was originally developed under a Cooperative Research and Development Agreement (CRADA) between the United States Department of Energy and three laboratories from private sector firms.

For more information about LAMMPS, visit the official LAMMPS website.

Programming language: C++
Brief description: open-source molecular dynamics program

Open-source license: GPL 2.0

Recommended Version

LAMMPS June 5, 2019

3.4.2 Environment Requirements

Hardware Requirements

Table 3-15 lists the hardware requirements.

Table 3-15 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-16 lists the software requirements.

Table 3-16 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAMMPS</td>
<td>June 5, 2019</td>
<td><a href="https://lammps.sandia.gov/tars/">https://lammps.sandia.gov/tars/</a></td>
</tr>
<tr>
<td>FFTW</td>
<td>3.3.8</td>
<td><a href="http://www.fftw.org/fftw-3.3.8.tar.gz">http://www.fftw.org/fftw-3.3.8.tar.gz</a></td>
</tr>
<tr>
<td>Test computing instance</td>
<td>in.lj</td>
<td>lammps-stable_5Jun2019/bench</td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-17 lists the OS requirements.

Table 3-17 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
3.4.3 Planning the Paths for Software Porting

Table 3-18 lists software installation paths involved in the LAMMPS software porting.

Table 3-18 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N/A</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/FFTW</td>
<td>Installation path of FFTW</td>
<td>The installation path listed in this table is only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/LAMMPS</td>
<td>Installation path of LAMMPS</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/CASE</td>
<td>Test path of LAMMPS</td>
<td></td>
</tr>
</tbody>
</table>

3.4.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 3-19 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install FFTW.</td>
<td>For details, see 3.4.4.1 Installing FFTW.</td>
</tr>
</tbody>
</table>
3.4.4.1 Installing FFTW

**Procedure**

**Step 1**
Use PuTTY to log in to the server as the root user.

**Step 2**
Run the following command to decompress the FFTW installation package:

```
tar -xvf fftw-3.3.8.tar.gz
```

**Step 3**
Run the following command to switch to the directory generated after decompression:

```
cd fftw-3.3.8
```

**Step 4**
Run the following command to perform configuration:

```
./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon
```

**Step 5**
Run the following commands to compile and install FFTW.

```
make -j 96
make install
```

**Step 6**
Run the following commands to load environment variables:

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

--- End

3.4.5 Obtaining the Source Code

**Procedure**

**Step 1**
Download the LAMMPS installation package `lammps-5Jun19.tar.gz` from:

[https://lammps.sandia.gov/tars/](https://lammps.sandia.gov/tars/)

**Step 2**
Use the SFTP tool to upload the LAMMPS installation package to the `/path/to/LAMMPS` directory on the server.

--- End

3.4.6 Compiling and Installing LAMMPS

**Procedure**

**Step 1**
Use PuTTY to log in to the server as the root user.

**Step 2**
Run the following command to switch to the directory, in which the LAMMPS installation package is stored:

```
cd /path/to/LAMMPS
```
**Step 3** Run the following command to decompress the LAMMPS installation package:

```
tar -xvf lammps-5Jun19.tar.gz
```

**Step 4** Run the following command to switch to the directory generated after decompression:

```
cd lammps-5Jun2019
```

**Step 5** Run the following command to switch to the `src` directory:

```
cd src
```

**Step 6** Perform the following operations to modify the `MAKE/OPTIONS/Makefile.g++_openmpi` file:

1. Run the `vi MAKE/OPTIONS/Makefile.g++_openmpi` command.
2. Press `I` to enter the insert mode and modify lines 54 to 56 in the `MAKE/OPTIONS/Makefile.g++_openmpi` file. Pay attention to the information in bold.
   
   ```
   FFT_INC = -DFFT_FFTW -I/path/to/FFTW/include
   FFT_PATH = -L/path/to/FFTW/lib
   FFT_LIB = -lfftw3
   ```

3. Press `Esc`, type `.wq!`, and press `Enter` to save the file and exit.

**Step 7** Run the following commands to perform compilation:

```
make yes-std
make no-lib
make -j 96 g++_openmpi
```

--- End

### 3.4.7 Running and Verifying LAMMPS

#### Procedure

**Step 1** Use PuTTY to log in to the server as the `root` user.

**Step 2** Run the following command to create the working directory:

```
mkdir -p path/to/CASE
```

**Step 3** Perform the following commands to switch to the working directory and copy the computing instance and binary files to the working directory.

```
cd /path/to/CASE

cp /path/to/LAMMPS/lammps-stable_5Jun2019/bench/in.lj ./
cp /path/to/LAMMPS/lammps-stable_5Jun2019/src/lmp_g++_openmpi ./
```

**Step 4** Run the following command:

```
mpirun --allow-run-as-root -np 96 --mca btl ^openib ./lmp_g++_openmpi -in in.lj
```

Check the value (in the unit of `timesteps/s`) of **Performance** in the `log.lammps` log file. A higher value indicates better performance.
The following is an example of the test result.

Performance: 593019.799 tau/day, 1372.731 timesteps/s
98.1% CPU use with 96 MPI tasks x no OpenMP threads

MPI task timing breakdown:

<table>
<thead>
<tr>
<th>Section</th>
<th>min time</th>
<th>avg time</th>
<th>max time</th>
<th>%varavg</th>
<th>%total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair</td>
<td>0.028215</td>
<td>0.040367</td>
<td>0.051175</td>
<td>3.4</td>
<td>55.41</td>
</tr>
<tr>
<td>Neigh</td>
<td>0.0036562</td>
<td>0.0049894</td>
<td>0.0061901</td>
<td>1.2</td>
<td>6.85</td>
</tr>
<tr>
<td>Comm</td>
<td>0.013968</td>
<td>0.026072</td>
<td>0.039602</td>
<td>4.8</td>
<td>35.79</td>
</tr>
<tr>
<td>Output</td>
<td>0.7905e-05</td>
<td>0.000128</td>
<td>0.00023195</td>
<td>0.0</td>
<td>0.18</td>
</tr>
<tr>
<td>Modify</td>
<td>0.00056847</td>
<td>0.00080538</td>
<td>0.00099514</td>
<td>0.0</td>
<td>1.11</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td>0.0004857</td>
<td></td>
<td></td>
<td>0.67</td>
</tr>
</tbody>
</table>

---End

3.5 Quantum ESPRESSO 6.4.1 Porting Guide (CentOS 7.6)

3.5.1 Introduction

Quantum ESPRESSO is a software suite for ab initio quantum chemistry methods for electronic-structure calculation and materials modeling, distributed for free and as free software free under the GNU General Public License. It is based on density functional theory (DFT), plane wave (PW) basis sets and pseudopotentials (including norm conservation and ultrasoft). Quantum ESPRESSO is an open initiative of the CNR-IOM DEMOCRITOS National Simulation Center in Trieste, Italy, and its partners, in collaboration with different centers worldwide such as MIT, Princeton University, the University of Minnesota or the Ecole Polytechnique Fédérale de Lausanne. The initiative is coordinated by the Quantum ESPRESSO Foundation, which is composed of a host of research centers and groups worldwide. Quantum ESPRESSO, written mainly in Fortran-90 with some parts in C or in Fortran-77, was built out of the merging and re-engineering of different independently-developed core packages, plus a set of packages designed to be inter-operable with the core components. The basic packages include Pwscf, which solves the self-consistent Kohn and Sham equations and obtains for a periodic solid, CP, which carries out Car-Parrinello molecular dynamics, and PostProc, which allows data analysis and plotting. Regarding the additional packages, it is noteworthy to point out atomic for the pseudopotential generation, PHonon package, which implements density-functional perturbation theory (DFPT) for the calculation of second- and third-order derivatives of the energy with respect to atomic displacements and NEB.

For more information about Quantum ESPRESSO, visit the Quantum ESPRESSO website.

Programming language: Fortran

Brief description: A software suite of ab initio quantum chemistry methods for electronic structure calculation and materials modeling.

Open-source protocol: GPL 2.0

Recommended Version

QE 6.4.1 is recommended.
3.5.2 Environment Requirements

Hardware Requirements

Table 3-20 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-21 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>QE</td>
<td>6.4.1</td>
<td><a href="https://github.com/QEF/q-e/tags">https://github.com/QEF/q-e/tags</a></td>
</tr>
<tr>
<td>OpenBLAS</td>
<td>0.3.7</td>
<td><a href="https://github.com/xianyi/OpenBLAS/tree/v0.3.6">https://github.com/xianyi/OpenBLAS/tree/v0.3.6</a></td>
</tr>
<tr>
<td>SCALAPACK</td>
<td>2.0.2</td>
<td><a href="http://www.netlib.org/scalapack/">http://www.netlib.org/scalapack/</a></td>
</tr>
<tr>
<td>Test case</td>
<td>AUSURF112</td>
<td><a href="https://github.com/QEF/benchmarks/tree/master/AUSURF112">https://github.com/QEF/benchmarks/tree/master/AUSURF112</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-22 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

3.5.3 Planning the Paths for Software Porting

Table Paths for software porting lists the software installation paths involved in the Quantum ESPRESSO software porting.
Table 3-23 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/SCALAPACK</td>
<td>Installation path of ScaLapack</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/QE</td>
<td>Installation path of Quantum ESPRESSO</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/CASE</td>
<td>Path for storing the test case plan of Quantum ESPRESSO</td>
<td></td>
</tr>
</tbody>
</table>

3.5.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Procedure

Table 3-24 Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see “Setting Up the Single-Node System Environment” in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install OpenBLAS.</td>
<td>For details, see 3.5.4.1 Installing OpenBLAS.</td>
</tr>
<tr>
<td>3</td>
<td>Install ScaLapack.</td>
<td>For details, see 3.5.4.2 Installing ScaLapack.</td>
</tr>
</tbody>
</table>
3.5.4.1 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following commands to decompress the OpenBLAS installation package:

```
tar xvf OpenBLAS-0.3.7.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
 cd OpenBLAS-0.3.7
```

Step 4 Run the following commands to declare the compilation environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 5 Run the following commands to perform compilation and installation:

```
 make
 make PREFIX=/path/to/OPENBLAS install
```

Step 6 Run the following command to load environment variables:

```
 export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

----End

3.5.4.2 Installing ScaLapack

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following commands to decompress the ScaLapack installation package:

```
tar -xvf scalapack.tgz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd scalapack-2.0.2
```

Step 4 Run the following command to generate the SLmake.inc file:

```
cp SLmake.inc.example SLmake.inc
```

Step 5 Run the following command to edit the SLmake.inc file:

1. vi SLmake.inc
2. Press i to enter the editing mode and modify lines 58 and 59 in the SLmake.inc file. Pay attention to the information in bold.

BLASLIB = -L/path/to/OPENBLAS/lib -lopenblas
LAPACKLIB = -L/path/to/OPENBLAS/lib -lopenblas
3. Press Esc, enter \texttt{.wq!}, and press Enter to save the file and exit.

**Step 6** Run the following command to perform compilation and installation:

```bash
make
```

**Step 7** Run the following command to copy the generated link library to the installation directory:

```bash
cp libscalapack.a /path/to/SCALAPACK
```

**Step 8** Run the following command to load environment variables:

```bash
export LD_LIBRARY_PATH=/path/to/SCALAPACK
```

---End

### 3.5.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the installation package \texttt{q-e-qe-6.4.1.tar.gz}.

URL: [https://github.com/QEF/q-e/tags](https://github.com/QEF/q-e/tags)

**Step 2** Use SFTP to upload the Quantum ESPRESSO installation package to the \texttt{/path/to/QE} directory on the server.

---End

### 3.5.6 Compiling and Installing Quantum ESPRESSO

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the Quantum ESPRESSO installation package:

```bash
cd /path/to/QE

tar -xvf q-e-qe-6.4.1.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:

```bash
cd q-e-qe-6.4.1
```

**Step 4** Run the following commands to set the blas, lapack, and scalapack libraries:

```bash
export BLAS_LIBS="-L/path/to/OPENBLAS/lib -lopenblas"
export LAPACK_LIBS="-L/path/to/OPENBLAS/lib -lopenblas"
export SCALAPACK_LIBS="-L/path/to/SCALAPACK -lscalapack"
```
Step 5 Run the following commands to perform configuration:
```
./configure F90=gfortran F77=gfortran MPIF90=mpifort MPIF77=mpifort
CC=mpicc \
FCFLAGS="-O3" CFLAGS="-O3" \
--with-scalapack=yes \
--prefix=/path/to/QE
```

Step 6 Run the following commands to perform compilation and installation:
```
make -j 20 pwall
make install
```

Step 7 Run the following command to load environment variables:
```
export PATH=/path/to/QE/bin:SPATH
```

3.5.7 Running and Verifying Quantum ESPRESSO

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to create and go to the working directory:
```
mkdir -p /path/to/CASE
cd /path/to/CASE
```

Step 3 Download the test case and use SFTP to upload the test case to the /path/to/CASE directory on the server.
URL: [https://github.com/QEF/benchmarks/tree/master](https://github.com/QEF/benchmarks/tree/master)

Step 4 Run the following commands to decompress the test case package and copy the required files to the /path/to/CASE directory:
```
unzip benchmarks-master.zip
cp benchmarks-master/AUSURF112/* ./
```

Step 5 Run the following command to create the hostfile file:

1. `vi hostfile`
2. Press `i` to enter the editing mode and add the following content:
   ```
   node1
   node2
   ```
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

Step 6 Open another login window and use PuTTY to log in to node 2 as the root user.

Step 7 On node 2, run the following command to configure environment variables:

1. `vi /root/.bashrc`
2. Press `i` to enter the editing mode, and add the following content to the end of the `.bashrc` file:
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
export PATH=/path/to/QE/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 8** On node 1, run the following command to start the Quantum ESPRESSO program:

```bash
mpirun --allow-run-as-root -np 192 -N 96 -x OMP_NUM_THREADS=1 -hostfile hostfile --mca btl ^openib pw.x -input ./ausurf.in 2>&1 | tee -a qe.log
```

<table>
<thead>
<tr>
<th>Table 3-25 Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter</strong></td>
</tr>
<tr>
<td>-np</td>
</tr>
<tr>
<td>-N</td>
</tr>
<tr>
<td>-x OMP_NUM_THREADS=1</td>
</tr>
<tr>
<td>--hostfile</td>
</tr>
</tbody>
</table>

If the information shown in Figure 3-1 is displayed at the end of the qe.log file generated by Quantum ESPRESSO, the Quantum ESPRESSO program ends properly.

Check the value of WALL in the qe.log file. The unit is second. A higher the value indicates a lower performance. Figure 3-1 shows the output result.

**Figure 3-1** Test example

```
This run was terminated on: 14:15:46 14Jan2020

WALL: 7m21.995s

JOB DONE.

Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG
PRIMARY JOB terminated normally, but 1 process returned a non-zero exit code. For user direction, the job has been aborted.

Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG
PRIMARY JOB terminated normally, but 1 process returned a non-zero exit code. For user direction, the job has been aborted.

----End
```
3.6 NAMD 2.13 Porting Guide (CentOS 7.6)

3.6.1 Introduction

Nanoscale Molecular Dynamics (NAMD) is a parallel molecular dynamics application that has been used to make breakthroughs in understanding the structure and dynamics of large biomolecular complexes. NAMD uses empirical force fields, such as Amber, CHARMM, and Dreiding, to calculate atomic trajectories by numerically solving motion equations. NAMD is used to predict the dynamic behavior and important properties of biomolecules, such as dispersive factor and cohesive energy.

For more information about NAMD, visit the official NAMD website.

Language: C

Brief description: The parallel molecular dynamics code for fast simulation of macromolecular systems.

Open-source protocol: user-defined open-source protocol

**Recommended Version**

NAMD 2.13

3.6.2 Environment Requirements

**Hardware Requirements**

*Table 3-26* lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

**Software Requirements**

*Table 3-27* lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAMD</td>
<td>2.13</td>
<td><a href="https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=NAMD">https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=NAMD</a></td>
</tr>
<tr>
<td>CHARM</td>
<td>6.9.0</td>
<td><a href="https://charm.cs.illinois.edu/distrib/">https://charm.cs.illinois.edu/distrib/</a></td>
</tr>
</tbody>
</table>
OS Requirements

Table 3-28 lists the OS requirements.

Table 3-28 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

3.6.3 Planning the Paths for Software Porting

Table 3-29 lists the software installation paths involved in the NAMD software porting.

Table 3-29 Paths for Software Porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/CHARM</td>
<td>Installation path of CHARM</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/FFTW</td>
<td>Installation path of FFTW</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/TCL</td>
<td>Installation path of TCL</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/NAMD</td>
<td>Installation path of NAMD</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/CASE</td>
<td>Path for storing the NAMD test case</td>
<td></td>
</tr>
</tbody>
</table>
3.6.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install CHARM.</td>
<td>For details, see Installing CHARM.</td>
</tr>
<tr>
<td>3</td>
<td>Install FFTW.</td>
<td>For details, see 3.6.4.1 Installing FFTW.</td>
</tr>
<tr>
<td>4</td>
<td>Install TCL.</td>
<td>For details, see Installing TCL.</td>
</tr>
</tbody>
</table>

3.6.4.1 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the FFTW installation package:

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd fftw-3.3.8
```

Step 4 Run the following command to run the bootstrap.sh script:

```
./bootstrap.sh
```

Step 5 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/FFTW --enable-threads --enable-openmp --enable-float
```

Step 6 Run the following commands to perform compilation and installation:

```
make -j40
make -j40 install
```
Step 7  Run the following commands to set the FFTW environment variables:

```
export PATH=/path/to/FFTW/bin:$PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

----End

3.6.5 Obtaining the Source Code

Procedure

Step 1  Download the NAMD installation package NAMD_2.13_Source.tar.gz.
   URL: [https://www.ks.uiuc.edu/Research/namd/](https://www.ks.uiuc.edu/Research/namd/)

Step 2  Use the SFTP tool to upload the NAMD installation package to the /path/to/NAMD directory on the server.

----End

3.6.6 Compiling and Installing NAMD

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to create the NAMD-related directories:

```
cd /path/to/NAMD
mkdir NAMD-2.13
cd NAMD-2.13
mkdir build install
export BUILD_DIR=`pwd`/build
export INSTALL_DIR=`pwd`/install
```

Step 3  Run the following commands to set the environment variables.

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 4  Run the following commands to compile and install CHARM:

```
cd /path/to/NAMD/NAMD-2.13
```

```
tar -xvf charm-6.9.0.tar.gz
cd charm-6.9.0
```

```
mkdir $(INSTALL_DIR)/charm690/
cp src/arch/mpi-linux-x86_64 src/arch/mpi-linux-arm8 -rf
```
grep -rl 'm64' src/arch/mpi-linux-arm8 | xargs sed -i 's/-m64//g'

Step 5 Run the following commands to decompress the NAMD installation package:

cd /path/to/NAMD/NAMD-2.13/build
tar -zxvf NAMD_2.13_Source.tar.gz

cd NAMD_2.13_Source

Step 6 Run the following command to create Linux-ARM64.fftw3:

cp arch/Linux-x86_64.fftw3 arch/Linux-ARM64.fftw3

Step 7 Run the following commands to upload and decompress the TCL file:

cd /path/to/NAMD/NAMD-2.13/build
tar -xvf tcl8.5.9-linux-arm64-threaded.tar.gz

Step 8 Run the following commands to modify the Make.charm file:

1. cd /path/to/NAMD/NAMD-2.13/build
2. vi Make.charm
3. Press i to enter the editing mode and modify the Make.charm file.
   CHARMBASE = /path/to/NAMD/NAMD-2.13/install/charm690

   **NOTE**
   Change the value of CHARMBASE to the actual installation path of the CHARM.

4. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 9 Run the following commands to modify the Linux-ARM64-g++.arch file:

1. cd /path/to/NAMD/NAMD-2.13/NAMD_2.13_Source
2. vi arch/Linux-ARM64-g++.arch
3. Press i to enter the editing mode and modify the Linux-ARM64-g++.arch file.
   NAMD_ARCH = Linux-ARM64
   CHARMARCH = mpi-linux-arm8
   #FLOATOPTS = -O2 -ffast-math -funsafe-math-optimizations -fomit-frame-pointer -march=armv7-a -mcpu=cortex-a9 -mtune=cortex-a9
   FLOATOPTS = -O3 -ffast-math -funsafe-math-optimizations -fomit-frame-pointer -march=armv8-a
   CXX = g++ -std=c++11
   CXXOPTS = $(FLOATOPTS)
   CXXNOALIASOPTS = $(FLOATOPTS) -fno-strict-aliasing
   CC = gcc
   COPTX = $(FLOATOPTS)

4. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 10 Run the following commands to configure NAMD:

./config Linux-ARM64-g++ --with-fftw3 --fftw-prefix /path/to/FFTW --with-tcl--tcl-prefix /path/to/NAMD/NAMD-2.13/build/tcl8.5.9-linux-arm64-threaded --charm-arch mpi-linux-arm8

cd Linux-ARM64-g++
Step 11 Run the following commands to compile and install NAMD:

```bash
make -j
```

Step 12 Run the following command to check whether an executable file is generated:

```bash
ll /path/to/NAMD/NAMD-2.13/build/NAMD_2.13_Source/Linux-ARM64-g++/namd2
```

```
-rwxr-xr-x 1 root root 16140248 Aug  7 15:50 namd2
```

-----End

3.6.7 Running and Verifying NAMD

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to set the environment variable for the NAMD executable file:

```bash
export PATH=/path/to/NAMD/NAMD-2.13/build/NAMD_2.13_Source/Linux-ARM64-g++:$PATH
```

Step 3 Go to the directory where the test case is stored.

```bash
cd /path/to/CASE
```

Step 4 Run the following command to decompress the case file:

```bash
tar -xf apoa1.tar.gz
```

Step 5 Run the following command to modify the case file:

```bash
1. vi apoa1/apoa1.namd
2. Press i to enter the editing mode and add the following content to the end of the apoa1/apoa1.namd file:
   FFTWEstimate yes
   FFTWUseWisdom no
3. Press Esc, enter :wq!, and press Enter to save the file and exit.
```

Step 6 Run the following commands to start a test:

```bash
cd apoa1
mpirun --allow-run-as-root --mca btl ^openib -np 1 namd2 +ppn 96 apoa1.namd +setcpuaffinity 2>&1 |tee -a namd.log
```

Check the value of days/ns in the last Info:Benchmark time in the namd.log file. The unit is days/ns. A smaller value indicates better performance. You can also convert the value to the value of ns/days. A higher value indicates better performance.

**Figure 3-2** shows an example of the test result.
3.6.8 More Information

Official Arm website:


3.7 VASP 5.4.4 Porting Guide (CentOS 7.6)

3.7.1 Introduction

Vienna Ab-initio Simulation Package (VASP) is a software package developed by the Hafner team of the University of Vienna for electronic structure calculation and quantum mechanics-molecular dynamics simulation. VASP is one of the most popular commercial software used in material simulation and computational physical science study.

For more information about MySQL, visit the official VASP website.

Language: Fortran

Brief description: electronic structure calculation and quantum mechanics-molecular dynamics simulation.

Open-source protocol: open-source commercial software.

Recommended Tool Version

VASP 5.4.4

3.7.2 Environment Requirements

Hardware Requirements

Table 3-31 lists the hardware requirements.
Table 3-31 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software requirements

Table 3-32 lists the software requirements.

Table 3-32 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>VASP</td>
<td>5.4.4</td>
<td><a href="https://www.vasp.at/">https://www.vasp.at/</a></td>
</tr>
<tr>
<td>FFTW</td>
<td>3.3.8</td>
<td><a href="http://www.fftw.org/download.html">http://www.fftw.org/download.html</a></td>
</tr>
<tr>
<td>OpenBLAS</td>
<td>0.3.6</td>
<td><a href="https://github.com/xianyi/OpenBLAS/tree/v0.3.6">https://github.com/xianyi/OpenBLAS/tree/v0.3.6</a></td>
</tr>
<tr>
<td>SCALAPACK</td>
<td>2.0.2</td>
<td><a href="http://www.netlib.org/scalapack/">http://www.netlib.org/scalapack/</a></td>
</tr>
<tr>
<td>VTST</td>
<td>179</td>
<td><a href="http://theory.cm.utexas.edu/vtsttools/download.html">http://theory.cm.utexas.edu/vtsttools/download.html</a></td>
</tr>
<tr>
<td>Wannier90</td>
<td>1.2</td>
<td><a href="http://www.wannier.org/code/wannier90-1.2.tar.gz">http://www.wannier.org/code/wannier90-1.2.tar.gz</a></td>
</tr>
<tr>
<td>vasp-small</td>
<td>Test case</td>
<td>-</td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-33 lists the OS requirements.

Table 3-33 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

3.7.3 Planning the Paths for Software Porting

Table 3-34 lists the software installation paths involved in the VASP software porting.
### Table 3-34 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each installation package for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/SCALAPACK</td>
<td>Installation path of ScaLapack</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/FFTW</td>
<td>Installation path of FFTW</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/VASP</td>
<td>Installation path of VASP</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/CASE</td>
<td>Test path of VASP</td>
<td></td>
</tr>
</tbody>
</table>

### 3.7.4 Configuring the Compilation Environment

**Prerequisites**

Use SFTP to upload the installation packages to the planned directories on the server.

**Configuration Process**

### Table 3-35 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Establishing the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing OpenBLAS.</td>
<td>For details, see 3.7.4.1 Installing OpenBLAS.</td>
</tr>
<tr>
<td>3</td>
<td>Installing ScaLapack</td>
<td>For details, see 3.7.4.2 Installing ScaLapack.</td>
</tr>
<tr>
<td>4</td>
<td>Installing FFTW</td>
<td>For details, see 3.7.4.3 Installing FFTW.</td>
</tr>
</tbody>
</table>

---

Copyright © Huawei Technologies Co., Ltd.
3.7.4.1 Installing OpenBLAS

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following commands to decompress the OpenBLAS installation package:

```bash
unzip OpenBLAS-develop.zip
```

**Step 3** Run the following command to switch to the directory containing the decompressed files:

```bash
cd OpenBLAS-develop
```

**Step 4** Run the following command to perform configuration:

```bash
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

**Step 5** Run the following commands to perform compilation and installation:

```bash
make
make PREFIX=/path/to/OPENBLAS install
```

----End

3.7.4.2 Installing ScaLapack

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following commands to decompress the ScaLapack installation package:

```bash
tar -xvf scalapack.tgz
```

**Step 3** Run the following command to switch to the directory containing the decompressed files:

```bash
cd scalapack-2.0.2
```

**Step 4** Run the following command to generate the `SLmake.inc` file:

```bash
cp SLmake.inc.example SLmake.inc
```

**Step 5** Run the following command to edit lines 58 and 59 of the `SLmake.inc` file:

1. Run the following command to edit the case file:

   ```bash
   vi SLmake.inc
   ```

2. Press `I` to enter the editing mode and modify lines 58 and 59 in the `SLmake.inc` file. Pay attention to the information in bold.

   ```bash
   BLASLIB = -L/path/to/OPENBLAS/lib -lopenblas
   LAPACKLIB = -L/path/to/OPENBLAS/lib -lopenblas
   ```

3. Press `Esc`, enter :wq!, and press `Enter` to save the script and exit.
Step 6 Run the following commands to perform compilation and installation:

```
make
```

Step 7 Run the following command to copy the generated link library to the installation directory:

```
mkdir -p /path/to/SCALAPACK
cp libscalapack.a /path/to/SCALAPACK
```

----End

3.7.4.3 Installing FFTW

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the FFTW installation package:

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to switch to the directory containing the decompressed files:

```
cd fftw-3.3.8
```

Step 4 Run the following command perform configuration:

```
./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon
```

Step 5 Run the following commands to perform compilation and installation:

```
make
make install
```

----End

3.7.5 Obtaining the Source Code

Procedure

Step 1 Download the VASP installation package `vasp.5.4.4.tar.gz`, VTSTcode package `vtstcode-179.tgz`, and WANNier90-1.2 installation package `wannier90-1.2`.

Download address: [https://www.vasp.at/](https://www.vasp.at/)

Download address: [http://theory.cm.utexas.edu/vtsttools/download.html](http://theory.cm.utexas.edu/vtsttools/download.html)

Download address: [http://www.wannier.org/code/wannier90-1.2.tar.gz](http://www.wannier.org/code/wannier90-1.2.tar.gz)

Step 2 Use SFTP to upload the WAVEWATCH III installation package to the `/path/to/VASP` directory on the server.

----End

3.7.6 Compiling and Installing the Software
3.7.6.1 Basic Installation

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the VSAP installation package:

```
cd /path/to/VASP

tar -xvf vasp.5.4.4.tar.gz
```

**Step 3** Run the following command to switch to the directory containing the decompressed files:

```
cd vasp.5.4.4
```

**Step 4** Run the following commands to set environment variables:

```
export PATH=/path/to/GNU/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:$LD_LIBRARY_PATH
export PATH=/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENMPI/lib:/path/to/OPENBLAS/lib:/path/to/SCALAPACK:/path/to/FFTW/lib:$LD_LIBRARY_PATH
```

**Step 5** Run the following command to create and edit the `makefile.include` file:

1. `vi makefile.include`
2. Press `I` to edit the `makefile.include` file. Pay attention to the information in bold.

```
##Precompiler options
CPP_OPTIONS= -DHOST="LinuxGNU" \
-DMPI -DMPI_BLOCK=8000 \ 
-Duse_collective \ 
-Dscalapack \ 
-DCACHE_SIZE=5000 \ 
-Ddavoidsflock \ 
-Duse_bse_te \ 
-Dbtbdyn \ 
-Duse_shmem

CPP        = gcc -E -P -w $*$(FUFFIX) >$*$(SUFFIX) $(CPP_OPTIONS)
FC         = mpif90
FCL        = mpif90
FREE       = -ffree-form -ffree-line-length-none
FFLAGS     = -w
OFLAG      = -O3 -ffp-contract=fast -fdec-math -ffpe-trap=invalid,zero,overflow -ffpe-summary=none
OFLAG_IN   = $(OFLAG)
DEBUG      = -O0
LLIBS      = -L/path/to/SCALAPACK -lscalapack -L/path/to/OPENBLAS/lib -lopenblas
FFTW       = -L/path/to/FFTW
LLIBS      = -L$(FFTW)/lib -lfftw3
INCS       = -I$(FFTW)/include
OBJECTS    = fftmpi.o fftmpi_map.o fftw3d.o fft3dlib.o
OBJECTS_O1 = fftw3d.o fftmpi.o fftmpiwi.o
OBJECTS_O2 = fft3dlib.o

# For what used to be vasp.5.lib
CPP_LIB    = $(CPP)
FC_LIB     = $(FC)
CC_LIB     = mpicc
CFLAGS_LIB = -O
FFLAGS_LIB = -O1
```

## Postcompiler links

```
CPP_LIB    = $(CPP)
FC_LIB     = $(FC)
CC_LIB     = mpicc
CFLAGS_LIB = -O
FFLAGS_LIB = -O1
```
FREE_LIB = $(FREE)
OBJECTS_LIB= linpack_double.o getshmem.o
# For the parser library
CXX_PARS   = g++
LIBS       += parser
LLIBS      += -Lparser -lparser -lstdc++
# Normally no need to change this
SRCDIR     = ../../src
BINDIR     = ../../bin

NOTE

Ensure that no space exists at the end of a line or after a backslash (\).

3. Press Esc, enter :wq!, and press Enter to save the script and exit.

Step 6 Run the following commands to perform compilation and installation:

make std
If you need to generate vaspgam and vaspncl, run the following command:

make all

----End

3.7.6.2 (Optional) Extended Installation

3.7.6.2.1 Installing the VTST Transition State Tools

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the VTSTcode installation package to the directory where the VASP is installed.

Step 3 Run the following commands to decompress the VTST installation package:

cd /path/to/VASP
tar -xzvf vtstcode-179.tgz

Step 4 Run the following command to back up the chain.F file in the src directory of the VASP installation package:

cp vasp-5.4.4/src/chain.F / vasp-5.4.4/src/chain.F_bak

Step 5 Run the following command to copy files in the vtstcode-179 directory to the src directory:

cp vtstconde-179/* vasp-5.4.4/src/

Step 6 Run the following commands to modify the src/main.F file:

1. Run the following command to edit the case file:

vi vasp-5.4.4/src/main.F

2. Press I to enter the editing mode and modify line 3147 in the SLmake.inc file. Pay attention to the information in bold.

Before the modification:
CALL CHAIN_FORCE(T_INFO%IONIS,DYN%POSION,TOTEN,TIFOR, & LATT_CUR%A,LATT_CUR%B,IO %IU6)

After the modification:
CALL CHAIN_FORCE(T_INFO%IONIS,DYN%POSION,TOTEN,TIFOR, & TSIF,LATT_CUR%A,LATT_CUR %B,IO%IU6)

3. Press Esc, enter :wq!, and press Enter to save the script and exit.

Step 7 Run the following command to modify the compilation configuration file:
1. Run the following command to edit the case file:
   vi vasp-5.4.4/src/.objects
2. Press I to enter the editing mode and add the following content between line 71 and line 72:
   bfgs.o dynmat.o instanton.o lbfgs.o sd.o cg.o dimer.o bbm.o \
   fire.o lanczos.o neb.o qm.o opt.o \

   [NOTE]
   Ensure that no space exists after a backslash (\). Replace spaces at the beginning of a line with tabs.
3. Press Esc, enter :wq!, and press Enter to save the script and exit.

Step 8 See Step 6 to perform compilation and installation.

   make all
   ----End

3.7.6.2.2 Installing the Wannier90 Interfaces

Wannier90 has built-in VASP interfaces. VASP 5.4.X can only be interconnected with Wannier90-v1.2.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the wannier90-1.2 installation package in the VASP installation directory:
   cd /path/to/VASP
   tar -xvf wannier90-1.2.tar.gz

Step 3 Run the following commands to replace make.sys with make.sys.gfort:
   cd wannier90-1.2
   cp config/make.sys.gfort make.sys

Step 4 Run the following command to modify the make.sys file:
1. Run the following command to modify the case file:
   vi make.sys
2. Press I to enter the editing mode and add the following line to the end of the make.sys file:
   LIBS = -L/path/to/SCALAPACK -L/path/to/OPENBLAS -lscalapack -lopenblas
3. Press Esc, enter :wq!, and press Enter to save the script and exit.
**Step 5** Run the following command to perform installation:

```
make wannier lib test
```

The `libwannier.a` function library is generated, which needs to be added to the VASP installation and compilation.

**Step 6** Modify the VASP compilation file `makefile.include`.

1. `cd /path/to/VASP/vasp-5.4.4`
2. `vi makefile.include`
3. Press i to enter the editing mode and modify the `makefile.include` file.
   
   Modify line line 10 as follows:
   ```
   -Duse_shmem -Dtbdyn -DVASP2WANNIER90
   ```
   
   Add the following information before line 24:
   ```
   WANNIER90 = /path/to/VASP/wannier90-1.2/libwannier.a
   ```
   
   Add the following interface parameter to `LLIBS`:
   ```
   LLIBS = $(WANNIER90) -L/path/to/SCALAPACK -lscalapack -L/path/to/SCALAPACK/lib -lopenblas
   ```
   
   4. Press Esc, enter :wq!, and press Enter to save the script and exit.

**Step 7** Run the following command to perform compilation:

```
make all
```

---End

### 3.7.7 Running and Verifying the Software

#### Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to create and go to the working directory:

```
mkdir -p /path/to/CASE
cd /path/to/CASE
```

**Step 3** Prepare the test case described in Table 3-32 and copy the case to the `/path/to/CASE` directory.

**Step 4** Run the following commands to set environment variables:

```
export PATH=/path/to/GNU/install/bin:SPATH
export LD_LIBRARY_PATH=/path/to/GNU/install/lib64:SDL_LIBRARY_PATH
export PATH=/path/to/OPENMPI/bin:/home/vasp/bin:SPATH
export LD_LIBRARY_PATH=/path/to/OPENMPI/lib:/path/to/OPENBLAS/lib:/path/to/SCALAPACK:/path/to/FFTW/lib:SDL_LIBRARY_PATH
```

**Step 5** Run the following command to create the `host` file:

1. `vi hostfile`
2. Press I to enter the editing mode and add the following content:
3. Press Esc, enter :wq!, and press Enter to save the script and exit.

**Step 6** Run the following command to start the VASP program.

```bash
cmpirun --allow-run-as-root -np 96 -x OMP_NUM_THREADS=1 --hostfile hostfile vasp_std 2>&1 | tee -a vasp.out
```

<table>
<thead>
<tr>
<th>Table 3-36 Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>-np</td>
</tr>
<tr>
<td>--hostfile</td>
</tr>
</tbody>
</table>

**Step 7** If the information shown in Figure 3-3 is displayed at the end of the OUTCAR log generated by VASP, the VASP program ends normally.

Check the value of real time in the OUTCAR log. The unit is second. A smaller value indicates better performance.

**Figure 3-3** shows the command output.

**Figure 3-3** Result example

```
LOOP: cpu time 2398.1089, real time 630.2001
40BIT: cpu time 0.0000, real time 0.0000

Total amount of memory used by VASP on root node 43360. kbytes

base : 30060. kbytes
molproj : 837. kbytes
fftplane : 5353. kbytes
grid : 7240. kbytes
one-center : 7. kbytes
wavefun : 127. kbytes

General timing and accounting informations for this job:

Total CPU time used [sec]: 2408.223
User time [sec]: 2407.093
System time [sec]: 0.330
Elapsed time [sec]: 641.775
Maximum memory used [kb]: 87424.
Average memory used [kb]: 0.
Minor page faults: 11558
Major page faults: 0
Voluntary context switches: 29297
```

----End

### 3.7.8 Support and Other Resources

Official VASP website: [https://www.vasp.at/](https://www.vasp.at/)

VASP manual: [https://cms.mpi.univie.ac.at/vasp/vasp/vasp.html](https://cms.mpi.univie.ac.at/vasp/vasp/vasp.html)
3.8 ABINIT 8.10.3 Porting Guide (CentOS 7.6)

3.8.1 Introduction

ABINIT is a software suite to calculate the optical, mechanical, vibrational, and other observable properties of materials. Starting from the quantum equations of density functional theory, you can build up to advanced applications with perturbation theories based on DFT, and many-body Green’s functions (GW and DMFT). ABINIT can calculate molecules, nanostructures and solids with any chemical composition, and comes with several complete and robust tables of atomic potentials.

For more information about ABINIT, visit the official ABINIT website.

Language: C/Fortran

Brief description: Software suite to calculate the optical, mechanical, vibrational, and other observable properties of materials

Open-source license: GPL 3.0

Recommended Version

ABINIT 8.10.3

3.8.2 Environment Requirements

Hardware Requirements

Table 3-37 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-38 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abinit</td>
<td>8.10.3</td>
<td><a href="https://www.abinit.org/packages">https://www.abinit.org/packages</a></td>
</tr>
<tr>
<td>AtomPAW</td>
<td>4.0.1.0</td>
<td><a href="https://www.abinit.org/fallbacks">https://www.abinit.org/fallbacks</a></td>
</tr>
</tbody>
</table>
### OS Requirements

Table 3-39 lists the OS requirements.

#### Table 3-39 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
</tbody>
</table>

### 3.8.3 Planning the Paths for Software Porting

Table 3-40 lists the software installation paths involved in the ABINIT software porting.

#### Table 3-40 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/PNETCDF</td>
<td>Installation path of Pnetcdf</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NetCDF-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installation path of NetCDF-Fortran</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/ABINIT</td>
<td>Installation path of ABINIT</td>
<td></td>
</tr>
</tbody>
</table>

### 3.8.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Configuration Process

**Table 3-41 Configuration process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>&quot;Setting Up the Environment for the Cluster Scenario&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Install OpenBLAS.</td>
<td>3.8.4.1 Installing OpenBLAS</td>
</tr>
<tr>
<td>3</td>
<td>Install Pnetcdf.</td>
<td>3.8.4.2 Installing Pnetcdf</td>
</tr>
<tr>
<td>4</td>
<td>Install HDF5.</td>
<td>3.8.4.3 Installing HDF5</td>
</tr>
<tr>
<td>5</td>
<td>Install NetCDF-C.</td>
<td>3.8.4.4 Installing NetCDF-C</td>
</tr>
<tr>
<td>6</td>
<td>Install NetCDF-Fortran.</td>
<td>3.8.4.5 Installing NetCDF-Fortran</td>
</tr>
</tbody>
</table>
3.8.4.1 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install BLAS LAPACK:

```bash
yum install blas64.aarch64 blas64-devel.aarch64 lapack64.aarch64 lapack64-devel.aarch64 -y
```

**NOTE**

If you use a non-default compiler to install the OpenBLAS, download the OpenBLAS and use the specified compiler to install it. Specify the OpenBLAS path for compilation and installation in Compiling and Installing ABINIT.

Step 3 Run the following command to decompress the OpneBLAS installation package:

```bash
tar xvf OpenBLAS-0.3.6.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd OpenBLAS-0.3.6
```

Step 5 Run the following command to perform configuration:

```bash
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 6 Run the following commands to perform compilation and installation:

```bash
make
make PREFIX=/path/to/OPENBLAS install
```

Step 7 Run the following commands to set OpenBLAS environment variables.

```bash
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

-----End

3.8.4.2 Installing Pnetcdf

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the Pnetcdf installation package:

```bash
tar -xvf parallel-netcdf-1.9.0.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd parallel-netcdf-1.9.0
```
Step 4 Run the following command to perform configuration:
```
./configure --prefix=/path/to/PNETCDF --build=aarch64-linux --enable-shared --enable-fortran --enable-large-file-test
```
Step 5 Run the following commands to perform compilation and installation:
```
make -j
make install
```
Step 6 Run the following commands to set the Pnetcdf environment variables:
```
export PATH=/path/to/PNETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PNETCDF/lib:$LD_LIBRARY_PATH
```

----End

3.8.4.3 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following commands to decompress the HDF5 installation package:
```
tar -xvf hdf5-1.10.1.tar.gz
```
Step 3 Run the following command to switch to the directory generated after the package is decompressed:
```
cd hdf5-1.10.1
```
Step 4 Run the following command to perform configuration:
```
./configure --prefix=/pat/to/HDF5 --build=aarch64-linux --enable-parallel --enable-shared --enable-fortran
```
Step 5 Run the following commands to perform compilation and installation:
```
make -j
make install
```
Step 6 Run the following commands to set the HDF5 environment variables:
```
export LD_LIBRARY_PATH=/pat/to/HDF5/lib:$LD_LIBRARY_PATH
```

----End

3.8.4.4 Installing NetCDF-C

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following commands to decompress the NetCDF-C installation package:
```
tar -zxvf netcdf-c-4.7.0.tar.gz
```
Step 3  Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-c-4.7.0
```

Step 4  Run the following command perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --enable-fortran --disable-static --enable-shared --with-pic --enable-parallel-tests --enable-pnetcdf --enable-large-file-tests --enable-largefile
```

Step 5  Run the following commands to perform compilation and installation:

```
make -j
make install
```

Step 6  Run the following commands to set the NetCDF-C environment variables:

```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
```

----End

### 3.8.4.5 Installing NetCDF-Fortran

**Procedure**

**Step 1**  Use PuTTY to log in to the server as the root user.

**Step 2**  Run the following commands to decompress the NetCDF-Fortran installation package:

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

**Step 3**  Run the following command to switch to the directory generated after the package is decompressed:

```
cd netcdf-fortran-4.4.5
```

**Step 4**  Run the following command to perform configuration:

```
./configure --prefix=/path/to/NETCDF --build=aarch64-linux --disable-static --enable-shared --enable-parallel-tests --enable-large-file-tests --enable-largefile
```

**Step 5**  Run the following commands to perform compilation and installation:

```
make -j
make install
```

----End
3.8.5 Obtaining the Source Code

Procedure

Step 1  Download the ABINIT installation package abinit-8.10.3.tar.gz.
URL: https://www.abinit.org/

Step 2  Use SFTP to upload the ABINIT installation package to the /path/to/ABINIT
directory on the server.

----End

3.8.6 Compiling and Installing ABINIT

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to set environment variables:

```
export MPI_HOME=$MPI_DIR
export CPPFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include -I/path/to/NETCDF/include -I$MPI_HOME/include"
export CFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include -I/path/to/NETCDF/include -I$MPI_HOME/include "
export CXXFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include -I/path/to/NETCDF/include -I$MPI_HOME/include "
export FCFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include -I/path/to/NETCDF/include -I$MPI_HOME/include "
export FFLAGS=" -I/path/to/HDF5/include -I/path/to/PNETCDF/include "
export LDFLAGS=" -L/path/to/HDF5/lib -L/path/to/PNETCDF/lib -L/path/to/NETCDF/lib"
```

Step 3  Run the following command to create an extral directory for storing third-party
math libraries:

```
  cd /path/to/ABINIT
  mkdir extral
```

Step 4  Upload the following software packages to the extral directory:

```
  atompaw-4.0.1.0.tar.gz
  bigdft-1.7.1.25.tar.gz
  libxc-3.0.0.tar.gz
  wannier90-2.0.1.1.tar.gz
```

Step 5  Run the following commands to decompress the ABINIT installation package:

```
  tar -zxvf abinit-8.10.3.tar.gz
```
Step 6 Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd abinit-8.10.3
```

Step 7 Run the following command to create and access the directory for compiling the ABINIT software:

```bash
mkdir build
cd build
```

Step 8 Run the following command perform configuration:

```bash
../configure
FC=mpif90 --enable-mpi --with-mpi-prefix=/path/to/OPENMPI/ --with-dft-flavor="libxc+atompaw" --with-trio-flavor="netcdf" --with-tardir=/path/to/ABINIT/external --with-netcdf-include="/path/to/NETCDF/include" --with-netcdf-libs="-L/path/to/NETCDF/lib -lnetcdff -L/path/to/NETCDF/lib -lnetcdff" FCFLAGS="-g -O2 -ffree-line-length-none" --with-linalg-libs="-L/path/to/OPENBLAS/ -lblas -llapack"
```

Step 9 Run the following commands to perform compilation and installation:

```bash
FC=mpif90 CC=mpicc CXX=mpicxx make multi multi_nprocs=96
```

Step 10 Run the following commands to add the environment variables:

```bash
export ABI_TESTS=/path/to/ABINIT/abinit-8.10.3/tests/
export ABI_TUTORIAL=$ABI_TESTS/tutorial/
export ABI_TUTORESPFN=$ABI_TESTS/tutorespfn/
export ABI_TUTOPARAL=$ABI_TESTS/tutoparal/
export ABI_TUTOPLUGS=$ABI_TESTS/tutoplugs/
export ABI_PSPDIR=$ABI_TESTS/Psps_for_tests/
export PATH=/path/to/ABINIT/abinit-8.10.3/build/src/98_main/:SPATH
```

----End

3.8.7 Running and Verifying Ceph

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to create a case directory and copy the case files:

```bash
cd SABI_TUTORIAL/Input
mkdir Work_paral
cd Work_paral
cp ../tbasepar_1.files .
cp ../tbasepar_1.in .
```
**Step 3** Run the following command:

```
abinit < tbasepar_1.files > log 2> err
```

**Step 4** Run the following commands:

1. Run the following command to create a folder:
   ```
   mkdir tmp
   ```

2. Run the following commands to add the test case packages:
   ```
   vi tbase1_x.files
   ```

3. Press `I` to enter the editing mode and create the test case files.
   ```
   tbasepar_1.in
   tbasepar_1.out
   tbasepar_1i
   tbasepar_1o
   ```
   `/tmp/tbasepar_1` # Change the value to a path that can be accessed by each node, for example, the created `tmp` path. You are advised to set the value to an absolute path.

   ```
   ../../Psps_for_tests/HGH/82pb.4.hgh
   ```

4. Press `Esc`, enter `.wq!`, and press `Enter` to save the file and exit.

5. Run the following command to perform configuration:
   ```
mpirun -np 2 ../../build/src/98_main/abinit < tbasepar_1.files >& tbasepar_1.log &
   ```

**NOTE**

Run the following command as user `root`:

```
mpirun --allow-run-as-root -np 2 ../../build/src/98_main/abinit < tbasepar_1.files >& tbasepar_1.log &
```

**Step 5** Check the value of **Real time** (in seconds) in the `tbasepar_1.log` log. A smaller value indicates better performance.

**Figure 3-4** shows the command output.

**Figure 3-4** Result example

---
Calculation completed.
Delivered 1 WARNINGs and 1 FOUNDs to log file.
---
Final Summary
program: abinit
version: 8.10.3
start datetime: Thu Aug 22 14:44:22 2019
end datetime: Thu Aug 22 14:44:27 2019
overall cpu time: 9.5
overall wall time: 9.7
exit requested by user: no
time limit: 0
pseudos:
Pb : 8b5eeb768eff7f35f37bcf891d99906
usepaw: 0
mpi_procs: 2
mpi_threads: 1
num_warnings: 1
num_comments: 1

-----End
3.8.8 Troubleshooting

Problem 1: An Error Is Reported When configure Is Run

Symptom
An error message "error: external libxc support does not work" is displayed when the configure command is executed.

Possible Causes
--with-libxc-libs= is not written in sequence.

Procedure
Add --with-libxc-libs=-lxcf90 -lxc--- after the command for Compiling and Installing ABINIT.

Problem 2: An Error Is Reported During ABINIT Compilation

Symptom
When the ABINIT is compiled, an error message "Error: Line truncated at (1) [Werror=line-truncation]" is displayed.

Possible Causes
This error occurs because the code does not limit the number of characters to 132.

Procedure
Add -ffree-line-length-none to the FCFLAGS="-g -O2 -ffree-line-length-none" command in Compiling and Installing ABINIT.

Problem 3: An Error Is Reported During ABINIT Compilation

Symptom
An error message "File'mpi.mod'opened at (1) is not a GNU Fortran module file" is displayed when you run the make command to compile ABINIT.

Possible Causes
This error occurs because the gcc and OpenMPI compiler versions do not match.

Procedure
Unifying the compilers used by OpenMPI and ABINIT can solve the problem.

NOTE
During the installation, the OpenMPI, BLAS-LAPACK, and NetCDF must use the same compiler. Otherwise, errors may occur.

Problem 4: An Error Is Reported During ABINIT Compilation

Symptom
When the ABINIT is compiled, an error message "abinit-8.10.3/config/gnu/missing: line 81: automake-1.16: command not found" is displayed.
Possible Causes

This error occurs because the aclocal and automake versions are set to 1.16 when the Makefile is generated.

Procedure

Check the automake version of the system and replace it with the system version in the Makefile.

**Step 1** vi abinit-8.10.3/build/Makefile

**Step 2** Press I to enter the insert mode and modify the system version in the Makefile file. Pay attention to the information in bold.

AUTOMAKE = ${SHELL} /storage/softwares/TaiShan/abinit/abinit-8.10.3/config/gnu/missing automake-1.16

**Step 3** Press Esc, enter :wq!, and press Enter to save the file and exit.

----End

Problem 5: An Error Is Reported When the Case Is Run

Symptom

An error is reported when the case is executed. The error information is similar to the segmentation fault:

```
[armmode=80976:0x80f79] Caught signal 11 [Segmentation fault: address not mapped to object at address 0x3c]
```

Possible Causes

The segmentation error is caused by the inconsistency between the compiler used to compile ABINIT and the compiler used to compile OpenMPI-4.0.1 and BLAS.

Procedure

Installing the AtomPAW can optimize the memory read/write of data.

**Step 1** Run the following command to access the directory storing the AtomPAW installation package:

```
cd /path/to/ABINIT/extral
```
Step 2  Run the following command to decompress the source code package:
  
  tar xvf atompaw-4.0.1.0.tar.gz

Step 3  Run the following command to go to source code directory:
  
  cd atompaw-4.0.1.0

Step 4  Run the following commands to perform compilation and installation:
  
  ./configure

  make

  make install

  ----End

3.8.9 More Information

ABINIT official installation guide:
  

Official installation guide of the ABINIT third-party link libraries:
  

3.9 AmberTools 19 Porting Guide (CentOS 7.6)

3.9.1 Introduction

AmberTools is a series of modeling, molecular mechanics, and dynamics simulation programs developed by professor Coleman at the University of California for biomolecules. There are many programs, such as building modules, which arrange solvent moisture molecules and perform charge fitting, and analysis tools can be used to perform NMR refinement and track analysis of dynamic calculations.

For details about AmberTools, visit the official AmberTools website.

Language: C, C++, Fortran

In one sentence: Molecular mechanics and dynamics simulation program.

Open-source license: GPL 3.0

Recommended Version

AmberTools 19

3.9.2 Environment Requirements

Hardware Requirements

Table 3-42 lists the hardware requirements.
### Table 3-42 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

### Software Requirements

Table 3-43 lists the software requirements.

#### Table 3-43 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>AmberTools</td>
<td>19</td>
<td><a href="http://ambermd.org/GetAmber.php#amber">http://ambermd.org/GetAmber.php#amber</a></td>
</tr>
<tr>
<td>Test case</td>
<td>nab</td>
<td>AmberTools 19 built-in cases</td>
</tr>
</tbody>
</table>

### OS Requirements

Table 3-44 lists the OS requirements.

#### Table 3-44 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 3.9.3 Planning the Paths for Software Porting

Table Paths for software porting lists the software installation paths involved in the AmberTools software porting.

#### Table 3-45 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/AMBER</td>
<td>Installation path of AmberTools</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

3.9.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-46 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see section &quot;Setting Up the Environment for the Cluster Scenario&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

3.9.5 Obtaining the Source Code

Procedure

**Step 1** Download the AmberTools installation package **AmberTools19.tar.bz2**.


**Step 2** Use SFTP to upload the AmberTools installation package to the /path/to/AMBER directory on the server.

----End

3.9.6 Compiling and Installing AmberTools

Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.
Step 2 Run the following commands to set environment variables:

```bash
export PATH=/path/to/GNU/bin:$PATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:$LD_LIBRARY_PATH
export PATH=/path/to/OPENMPI/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENMPI/lib:$LD_LIBRARY_PATH
```

Step 3 Run the following commands to decompress the AmberTools installation package:

```bash
cd /path/to/AMBER

tar -xvf AmberTools19.tar.bz2
```

Step 4 Run the following commands to update the `config.guess` and `config.sub` scripts in the `fftw3-3.3` directory:

```bash
cd /path/to/AMBER/amber18/AmberTools/src/fftw-3.3

http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD

vi config.guess

http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD

vi config.sub
```

Step 5 Run the following commands to update the `config.guess` and `config.sub` scripts in the `xblas` directory:

```bash
cd /path/to/AMBER/amber18/AmberTools/src/xblas/config

http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD

vi config.guess

http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD

vi config.sub
```

Step 6 Run the following commands to perform serial compilation and installation:

```bash
cd /path/to/AMBER/amber18

./configure --skip-python -nosse gnu

test -f /path/to/AMBER/amber18/amber.sh && source /path/to/AMBER/amber18/amber.sh

make install
```

Step 7 Run the following command to perform a serial test:

```bash
make test
```

Step 8 Run the following commands to perform parallel compilation and installation:

```bash
cd /path/to/AMBER/amber18
```
3.9.7 Running and Verifying the AmberTools

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to go to the case directory of AmberTools 18:

```
cd /path/to/AMBER/amber18/AmberTools/benchmarks/nab
```

Step 3  Run the following commands to modify the bench_amber file:

1. `vi bench_amber`
2. Press `I` to enter the editing mode and modify the bench_amber file.

   ```
   #!/bin/sh
   sander=sander.MPI
   cat<<EOF > mdin
   benchmark a short md
   &cntrl
   igb=1, cut=20.0, rgbmax=20.0,
   tempi=50.0, temp0=100.0, taup=0.4, ntt=1,
   ntb=0, nstlim=1000, ntpr=10,
   ntx=1, ireset=0,
   ntc=2, ntf=2, tol=0.0000001,
   /
   EOF
   $DO_PARALLEL $sander -O -i mdin -p $1.top -c $1.mc.x -o $1.md.o
   /bin/rm mdin restrt mdinfo
   ```
   3. Press `Esc`, enter `.wq!`, and press `Enter` to save the file and exit.

Step 4  Run the following command to perform parallel computing:

```
./bench_amber halfam0
```

Step 5  View the ns/day value in the halfam0.md.o log file. A larger value indicates better performance.

The following is an example of the output result.
### 3.9.8 Troubleshooting

**Problem 1: An Error Is Reported When configure Is Run**

**Symptom:**

An error message “Error: FFTW configure returned 1” is displayed when you run the `configure` command.

**Possible causes:**

- The FFTW3 certificate provided by AmberTools has expired.
- The `-nosse` parameter is not added when the `configure` command is executed.

**Procedure:**

Obtain the latest `config.guess` and `config.sub` files based on the website displayed in the error message. Run the `vi` command to update the `config.guess`
and config.sub files of FFTW3. Add the -nosse parameter when running the configure command.

3.9.9 More Information

Official AmberTools website:

http://ambermd.org/index.php

3.10 CP2K 4.1 Porting Guide (CentOS 7.6)

3.10.1 Introduction

CP2K is a quantum chemistry and solid-state physics package that can perform atomic simulations of solid state, liquid, molecular, periodic, material, crystal, and biological systems. It provides a general framework for different modeling methods. It supports DFTB, LDA, GGA, MP2, RPA, semi-empirical methods (AM1, PM3, PM6, RM1, and MNDO), and classical force fields (AMBER and CHARMM). CP2K can use NEB or dimer method to do simulations of molecular dynamics, metadynamics, Monte Carlo, Ellenfast dynamics, vibration analysis, core level spectroscopy, energy minimization, and transition state optimization.

For more information about CP2K, visit the official CP2K website.

Programming language: Fortran 2008

Brief description: a quantum chemistry and solid-state physics package used to perform atomic simulations of solid state, liquid, molecular, periodic, material, crystal, and biological systems

Open-source license: GPL

Recommended Versions

cp2k-4.1

3.10.2 Environment Requirements

Hardware Requirements

Table 3-47 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-48 lists the software requirements.
### Table 3-48 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP2K</td>
<td>4.1</td>
<td><a href="https://github.com/cp2k/cp2k/releases/">https://github.com/cp2k/cp2k/releases/</a></td>
</tr>
<tr>
<td>OpenBLAS</td>
<td>0.3.6</td>
<td><a href="https://github.com/xianyi/OpenBLAS/releases">https://github.com/xianyi/OpenBLAS/releases</a></td>
</tr>
<tr>
<td>Libint</td>
<td>1.1.5</td>
<td><a href="https://github.com/evaleev/libint">https://github.com/evaleev/libint</a></td>
</tr>
<tr>
<td>libXC</td>
<td>3.0.0</td>
<td><a href="https://www.abinit.org/fallbacks">https://www.abinit.org/fallbacks</a></td>
</tr>
<tr>
<td>FFTW</td>
<td>3.3.8</td>
<td><a href="https://www.cp2k.org/static/downloads/">https://www.cp2k.org/static/downloads/</a></td>
</tr>
<tr>
<td>LAPACK</td>
<td>3.8.0</td>
<td><a href="https://www.cp2k.org/static/downloads/">https://www.cp2k.org/static/downloads/</a></td>
</tr>
<tr>
<td>ScaLAPACK</td>
<td>2.1.0</td>
<td><a href="https://www.cp2k.org/static/downloads/">https://www.cp2k.org/static/downloads/</a></td>
</tr>
<tr>
<td>Test computing instance</td>
<td>H2O-64.inp</td>
<td>Provided by the software.</td>
</tr>
</tbody>
</table>

### OS Requirements

**Table 3-49** lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 3.10.3 Planning the Paths for Software Porting

**Table 3-50** lists the software installation paths involved in the CP2K software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------</td>
<td>-------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the porting process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/LIBINT</td>
<td>Installation path of Libint</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/FFTW</td>
<td>Installation path of FFTW</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/LIBXC</td>
<td>Installation path of Libxc</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/SCALAPACK</td>
<td>Installation path of ScaLAPACK</td>
<td></td>
</tr>
</tbody>
</table>

### 3.10.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

#### Configuration Process

**Table 3-51 Configuration process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>&quot;Setting Up the Single-Node System Environment&quot; in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>Install OpenBLAS.</td>
<td><a href="#">3.10.4.1 Installing OpenBLAS</a></td>
</tr>
<tr>
<td>3</td>
<td>Install Libint.</td>
<td><a href="#">3.10.4.2 Installing Libint</a></td>
</tr>
<tr>
<td>4</td>
<td>Install FFTW.</td>
<td><a href="#">3.10.4.3 Installing FFTW</a></td>
</tr>
<tr>
<td>5</td>
<td>Install Libxc.</td>
<td><a href="#">3.10.4.4 Installing Libxc</a></td>
</tr>
<tr>
<td>6</td>
<td>Install ScaLAPACK.</td>
<td><a href="#">3.10.4.5 Installing ScaLAPACK</a></td>
</tr>
</tbody>
</table>
3.10.4.1 Installing OpenBLAS

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to install BLAS LAPACK:

```bash
yum install blas64.aarch64 blas64-devel.aarch64 lapack64.aarch64 lapack64-devel.aarch64 -y
```

**NOTE**
If you use a non-default compiler to install OpenBLAS, download OpenBLAS and use the specified compiler to install it. Specify the path for OpenBLAS compilation and installation in **Step 3**.

**Step 3** Decompress the OpenBLAS installation package.

```bash
tar -xvf OpenBLAS-0.3.6.tar.gz
```

**Step 4** Run the following commands to perform the configuration:

```bash
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

**Step 5** Run the following commands to compile and install OpenBLAS.

```bash
make
make PREFIX=/path/to/OPENBLAS/ install
```

----End

3.10.4.2 Installing Libint

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Decompress the Libint installation package.

```bash
tar -xvf libint-1.1.5.tar.gz
```

**Step 3** Switch to the folder generated after decompression.

```bash
cd libint-1.1.5
```

**Step 4** Run the following command to perform the configuration:

```bash
./configure --prefix=/path/to/LIBINT/ -with-ar=ar FC=gfortran F77=gfortran F90=gfortran CC=gcc
```

**Step 5** Run the following commands to compile and install Libint:

```bash
make -j
```
make install

3.10.4.3 Installing FFTW

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Decompress the FFTW installation package.

```bash
  tar -zxvf fftw-3.3.8.tar.gz
```

Step 3  Switch to the folder generated after decompression.

```bash
  cd fftw-3.3.8
```

Step 4  Run the following command to perform the configuration:

```bash
  ./configure --prefix=/path/to/FFTW CC=gcc FC=gfortran F77=gfortran --enable-threads --enable-shared=yes
```

Step 5  Run the following commands to compile and install FFTW:

```bash
  make -j
  make install
```

3.10.4.4 Installing Libxc

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Decompress the Libxc installation package.

```bash
  tar zxfv libxc-3.3.0.tar.gz
```

Step 3  Switch to the folder generated after decompression.

```bash
  cd libxc-3.3.0
```

Step 4  Run the following command to perform the configuration:

```bash
  ./configure --prefix=/path/to/LIBXC AR=ar FC=gfortran F77=gfortran F90=gfortran CC=gcc
```

Step 5  Run the following commands to compile and install Libxc:

```bash
  make -j
  make install
```
3.10.4.5 Installing ScaLAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Decompress the ScaLAPACK installation package.
   
   tar –xvf scalapack-2.1.0.tgz

Step 3 Switch to the folder generated after decompression.
   
   cd scalapack-2.1.0

Step 4 Run the following command to copy and paste the file:
   
   cp SLmake.inc.example SLmake.inc

Step 5 Modify the SLmake.inc file.
1. Run vi SLmake.inc to open the file.
2. Press I to enter the editing mode and create the computing instance files.
   
   BLASLIB    = /path/to/OPENBLAS/lib/libopenblas.so
   LAPACKLIB  = /path/to/OPENBLAS/lib/libopenblas.so
   LIBS       = $(LAPACKLIB) $(BLASLIB)

3. Press Esc, type :wq!, and press Enter to save the file and exit.

Step 6 Run the following command to compile ScaLAPACK:
   
   make

----End

3.10.5 Obtaining the Source Code

Procedure

Step 1 Download the CP2K installation package cp2k-4.1.tar.bz2.
   
   URL: https://github.com/cp2k/cp2k/releases/

Step 2 Use the SFTP tool to upload the CP2K installation package to the /path/to/CP2K directory on the server.

----End

3.10.6 Compiling and Installing CP2K

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Decompress the CP2K installation package.
   
   tar xvf cp2k-4.1.tar.bz2

Step 3 Switch to the folder generated after decompression.
   
   cd cp2k-4.1/arch
**Step 4** Modify the configuration file.

1. Run the `vi Linux-arm-64-gfortran.psm` command to open the file.
2. Press `I` to enter the editing mode and create the computing instance files.

   ```
   FFTW_INC = /path/to/FFTW/fftw-3.3.8/include
   FFTW_LIB = /path/to/FFTW/fftw-3.3.8-build/lib
   LIBINT_INC = /path/to/LIBINT/include
   LIBINT_LIB = /path/to/LIBINT/lib
   LIBXC_INC = /path/to/LIBXC/libxc-3.0.0/include
   LIBXC_LIB = /path/to/LIBXC/libxc-3.0.0/lib
   OPENBLAS_INC = /path/to/OPENBLAS/OpenBLAS-0.3.6/include
   OPENBLAS_LIB = /path/to/OPENBLAS/OpenBLAS-0.3.6/lib
   DFLAGS = -D_FFTW3 -D_LIBINT -D_LIBXC2
   -D_LIBINT_MAX_AM=7 -D_LIBDERIV_MAX_AM=6
   -D_MAX_CONTR=4
   -D_para11e1 -D_SCALAPACK
   CPPFLAGS =
   FCFLAGS = $(DFLAGS) -O2 -ffast-math -ffree-form -ffree-line-length-none
   -mtune=native
   -I$(FFTW_INC) -I$(OPENBLAS_INC) -I$(LIBINT_INC) -I$(LIBXC_INC)
   -L/path/to/SCALAPACK/lib
   LDFLAGS = $(FCFLAGS) -static-libgfortran
   LIBS = $(OPENBLAS_LIB)/libopenblas.a
   $(FFTW_LIB)/libfftw3.a
   $(FFTW_LIB)/libfftw3_threads.a
   $(LIBXC_LIB)/libxcf90.a
   $(LIBXC_LIB)/libxc.a
   $(LIBINT_LIB)/libderiv.a
   $(LIBINT_LIB)/libint.a
   ``

3. Press `Esc`, type `:wq!`, and press `Enter` to save the file and exit.

**Step 5** Run the following commands to compile and install CP2K.

```
cd /path/to/CP2K/cp2k-4.1/makefiles
make -j 8 ARCH=Linux-arm-64-gfortran VERSION=psmp
```

----End

### 3.10.7 Running and Verifying CP2K

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to perform parallel computing:

1. Load environment variables.

   ```
   export PATH=/path/to/CP2K/cp2k-4.1/exe/Linux-x86-64-gfortran:SPATH
   ```

2. Run the following command as the root user:

   ```
   cd /path/to/CP2K/cp2k-4.1/tests/QS/benchmark
   mpirun --allow-run-as-root -np 96 -x OMP_NUM_THREADS=1 cp2k.psm
   H2O-64.inp > cp2k.H2O-64.inp.log
   ```

   The time difference between the output examples (1) and (2) is 8095.77s.

**Figure 3-6 Output example (1)
3.10.8 Troubleshooting

An Error Is Reported When CP2K Is Run

**Symptom**

When `cp2k.psmp` is run, “mpirun noticed that process rank 0 with PID 0 on node XA320V2-90 exited on signal 11 (Segmentation fault)” is reported.

**Possible Causes**

The software is memory consuming, causing shortage of memory resources.

**Procedure**

Increase the memory capacity or use a small computing instance provided by the software.

3.10.9 More Information

CP2K installation guide:

[https://www.cp2k.org/howto:compile](https://www.cp2k.org/howto:compile)

3.11 CP2K 7.1 Porting Guide (CentOS 7.6)

3.11.1 Introduction

CP2K is a quantum chemistry and solid-state physics package that can perform atomic simulations of solid state, liquid, molecular, periodic, material, crystal, and biological systems. It provides a general framework for different modeling methods. It supports DFTB, LDA, GGA, MP2, RPA, semi-empirical methods (AM1, PM3, PM6, RM1, and MNDO), and classical force fields (AMBER and CHARMM). CP2K can use NEB or dimer method to do simulations of molecular dynamics, metadynamics, Monte Carlo, Ellenfast dynamics, vibration analysis, core level spectroscopy, energy minimization, and transition state optimization.

For more information about CP2K, visit the [official CP2K website](https://www.cp2k.org)

**Programming language:** Fortran 2008

**Brief description:** a quantum chemistry and solid-state physics package used to perform atomic simulations of solid state, liquid, molecular, periodic, material, crystal, and biological systems

**Open-source license:** GPL
Recommended Version

CP2k-7.1

3.11.2 Environment Requirements

Hardware Requirements

Table 3-52 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-53 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP2K</td>
<td>7.1</td>
<td><a href="https://github.com/cp2k/cp2k/archive/v7.1.0.tar.gz">https://github.com/cp2k/cp2k/archive/v7.1.0.tar.gz</a></td>
</tr>
<tr>
<td>Libint</td>
<td>2.6.0</td>
<td><a href="https://github.com/evaleev/libint/archive/v2.6.0.tar.gz">https://github.com/evaleev/libint/archive/v2.6.0.tar.gz</a></td>
</tr>
<tr>
<td>Libxc</td>
<td>4.3.4</td>
<td><a href="https://www.abinit.org/fallbacks">https://www.abinit.org/fallbacks</a></td>
</tr>
<tr>
<td>FFTW</td>
<td>3.3.8</td>
<td><a href="https://www.cp2k.org/static/downloads/">https://www.cp2k.org/static/downloads/</a></td>
</tr>
<tr>
<td>LAPACK</td>
<td>3.8.0</td>
<td><a href="https://www.cp2k.org/static/downloads/">https://www.cp2k.org/static/downloads/</a></td>
</tr>
<tr>
<td>ScaLAPACK</td>
<td>2.1.0</td>
<td><a href="https://www.cp2k.org/static/downloads/">https://www.cp2k.org/static/downloads/</a></td>
</tr>
<tr>
<td>CMake</td>
<td>3.16.4</td>
<td><a href="https://cmake.org/files/v3.16/cmake-3.16.4.tar.gz">https://cmake.org/files/v3.16/cmake-3.16.4.tar.gz</a></td>
</tr>
<tr>
<td>GMP</td>
<td>6.2.0</td>
<td><a href="https://gmplib.org/download/gmp/gmp-6.2.0.tar.xz">https://gmplib.org/download/gmp/gmp-6.2.0.tar.xz</a></td>
</tr>
<tr>
<td>Boost</td>
<td>1.72</td>
<td><a href="https://dl.bintray.com/boostorg/release/1.72.0/source/boost_1_72_0.tar.gz">https://dl.bintray.com/boostorg/release/1.72.0/source/boost_1_72_0.tar.gz</a></td>
</tr>
<tr>
<td>DBCSR</td>
<td>2.0.1</td>
<td><a href="https://github.com/cp2k/dbcsr/releases/download/v2.0.1/dbcsr-2.0.1.tar.gz">https://github.com/cp2k/dbcsr/releases/download/v2.0.1/dbcsr-2.0.1.tar.gz</a></td>
</tr>
<tr>
<td>ELPA</td>
<td>019.05.001</td>
<td><a href="https://www.cp2k.org/static/downloads/elpa-2019.05.001.tar.gz">https://www.cp2k.org/static/downloads/elpa-2019.05.001.tar.gz</a></td>
</tr>
</tbody>
</table>
### OS Requirements

Table 3-54 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 3.11.3 Planning the Paths for Software Porting

Table 3-55 lists the software installation paths involved in the CP2K software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake.</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/GMP</td>
<td>Installation path of GMP.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/BOOST</td>
<td>Installation path of Boost.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/LIBINT</td>
<td>Installation path of Libint.</td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Usage</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>6</td>
<td>/path/to/FFTW</td>
<td></td>
<td>Installation path of FFTW.</td>
</tr>
<tr>
<td>7</td>
<td>/path/to/LAPACK</td>
<td></td>
<td>Installation path of LAPACK.</td>
</tr>
<tr>
<td>8</td>
<td>/path/to/SCALAPACK</td>
<td></td>
<td>Installation path of ScaLAPACK.</td>
</tr>
<tr>
<td>9</td>
<td>/path/to/ELPA</td>
<td></td>
<td>Installation path of ELPA.</td>
</tr>
<tr>
<td>10</td>
<td>/path/to/SPGLIB</td>
<td></td>
<td>Installation path of Spglib.</td>
</tr>
<tr>
<td>11</td>
<td>/path/to/LIBXC</td>
<td></td>
<td>Installation path of Libxc.</td>
</tr>
<tr>
<td>12</td>
<td>/path/to/GSL</td>
<td></td>
<td>Installation path of GSL.</td>
</tr>
<tr>
<td>13</td>
<td>/path/to/PLUMED</td>
<td></td>
<td>Installation path of PLUMED.</td>
</tr>
<tr>
<td>14</td>
<td>/path/to/EXTRA</td>
<td></td>
<td>Installation paths for storing the library files generated by Libint, FFTW, LAPACK, ScaLAPACK, ELPA, Spglib, Libxc, GSL and PLUMED.</td>
</tr>
</tbody>
</table>

### 3.11.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Configuration Process

**Table 3-56** Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>No.</td>
<td>Operation</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>2</td>
<td>Installing CMake.</td>
<td>For details, see <a href="#">3.11.4.1 Installing CMake</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Installing GMP.</td>
<td>For details, see <a href="#">3.11.4.2 Installing GMP</a>.</td>
</tr>
<tr>
<td>4</td>
<td>Installing Boost.</td>
<td>For details, see <a href="#">3.11.4.3 Installing Boost</a>.</td>
</tr>
<tr>
<td>5</td>
<td>Installing Libint.</td>
<td>For details, see <a href="#">3.11.4.4 Installing Libint</a>.</td>
</tr>
<tr>
<td>6</td>
<td>Installing FFTW.</td>
<td>For details, see <a href="#">3.11.4.5 Installing FFTW</a>.</td>
</tr>
<tr>
<td>7</td>
<td>Installing LAPACK.</td>
<td>For details, see <a href="#">3.11.4.6 Installing LAPACK</a>.</td>
</tr>
<tr>
<td>8</td>
<td>Installing ScaLAPACK.</td>
<td>For details, see <a href="#">3.11.4.7 Installing ScaLAPACK</a>.</td>
</tr>
<tr>
<td>9</td>
<td>Installing ELPA.</td>
<td>For details, see <a href="#">3.11.4.8 Installing ELPA</a>.</td>
</tr>
<tr>
<td>10</td>
<td>Installing Spglib.</td>
<td>For details, see <a href="#">3.11.4.9 Installing Spglib</a>.</td>
</tr>
<tr>
<td>11</td>
<td>Installing Libxc.</td>
<td>For details, see <a href="#">3.11.4.10 Installing Libxc</a>.</td>
</tr>
<tr>
<td>12</td>
<td>Installing GSL.</td>
<td>For details, see <a href="#">3.11.4.11 Installing GSL</a>.</td>
</tr>
<tr>
<td>13</td>
<td>Installing PLUMED.</td>
<td>For details, see <a href="#">3.11.4.12 Installing PLUMED</a>.</td>
</tr>
</tbody>
</table>

### 3.11.4.1 Installing CMake

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the CMake installation package:

```bash
tar -xvf cmake-3.16.4.tar.gz
```

**Step 3** Run the following command to decompress the CMake installation package:

```bash
cd cmake-3.16.4
```

**Step 4** Run the following command to perform configuration:

```bash
./configure --prefix=/path/to/CMAKE
```
Step 5 Run the following command to perform compilation and installation:

```
make -j&&make install
```

Step 6 Run the following command to set the environment variable:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

```
----End
```

### 3.11.4.2 Installing GMP

Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to decompress the GMP installation package:

```
tar -xvf gmp-6.2.0.tar.xz
```

**Step 3** Run the following commands to perform configuration:

```
cd gmp-6.2.0
./configure --prefix=/path/to/GMP
```

**Step 4** Run the following command to perform compilation and installation:

```
make -j&&make install
```

```
----End
```

### 3.11.4.3 Installing Boost

Procedure

**Step 1** Run the following command to decompress the Boost installation package:

```
tar -xvf boost_1_72_0.tar.gz
```

**Step 2** Run the following commands to perform configuration:

```
./bootstrap.sh
```

**Step 3** Run the following command to perform compilation and installation:

```
./b2 install --prefix=/path/to/BOOST
```

```
----End
```

### 3.11.4.4 Installing Libint

Procedure

**Step 1** Run the following command to install the dependency package using Yellowdog Updater, Modified (YUM):

```
yum install gmp-devel.aarch64 libudev* -y
```
Step 2 Run the following commands to create the installation directory:

```bash
mkdir -p /path/to/EXTRA
mkdir -p /path/to/EXTRA/mathlib
```

Step 3 Run the following command to decompress the Libint installation package:

```bash
tar -xvf libint-2.6.0.tar.gz
```

Step 4 Run the following commands to perform configuration:

```bash
cd libint-2.6.0
./autogen.sh
```

Step 5 Run the following commands to switch to the directory generated after decompression:

```bash
mkdir build
cd build
```

Step 6 Run the following commands to switch to the directory generated after decompression:

```bash
export LDFLAGS="-L/path/to/GMP/gmp-6.2/lib -L/path/to/BOOST/boost_1_72/lib"
export CPPFLAGS="-I/path/to/BOOST/boost_1_72/include/ -I/path/to/GMP/gmp-6.2/include"
```

Step 7 Run the following command to perform configuration:

```bash
../configure CXX=mpicxx --enable-eri=1 --enable-eri2=1 --enable-eri3=1 --with-max-am=4 --with-eri-max-am=4,3 --with-eri2-max-am=6,5 --with-eri3-max-am=6,5 --with-opt-am=3 --enable-generic-code --disable-unrolling --with-libint-exportdir=libint_cp2k_lmax4
```

Step 8 Run the following command to perform compilation:

```bash
make export
```

Step 9 Run the following commands to decompress the `libint_cp2k_lmax4.tgz` installation package.

```bash
tar -xvf libint_cp2k_lmax4.tgz
cd libint_cp2k_lmax4
```

Step 10 Run the following command to perform configuration:

```bash
./configure --prefix=/path/to/EXTRA/libint2 CC=mpicc CXX=mpicxx FC=mpifort
--enable-fortran --enable-shared
```

Step 11 Run the following commands to perform compilation and installation:

```bash
make -j 32
make install
```
3.11.4.5 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the FFTW installation package:

```
tar -zxvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd fftw-3.3.8
```

Step 4 Run the following command to perform configuration:

```
./configure CC=gcc F77=gfortran --enable-shared --enable-threads --enable-openmp --enable-mpi MPICC=mpicc --prefix=/path/to/EXTRA/fftw3
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

3.11.4.6 Installing LAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the LAPACK installation package:

```
tar zxvf lapack-3.8.0.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cp make.inc.example make.inc
```

Step 4 Run the following command to perform compilation and installation:

```
make -j
```

Step 5 Run the following command to copy the static library to another directory:

```
 cp *.a /path/to/EXTRA/mathlib
```

3.11.4.7 Installing ScaLAPACK

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following command to decompress the ScaLAPACK installation package:

```
tar -xvf scalapack-2.1.0.tgz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd scalapack-2.1.0
```

Step 4 Run the following command to copy and paste the file:

```
cp SLmake.inc.example SLmake.inc
```

Step 5 Run the following commands to modify the `SLmake.inc` file:

1. `vim SLmake.inc`
2. Press `i` to enter the editing mode.

   ```
   BLASLIB = /path/to/LAPACK/lapack-3.8.0/librefblas.a
   LAPACKLIB = /path/to/LAPACK/lapack-3.8.0/liblapack.a
   LIBS = $(LAPACKLIB) $(BLASLIB)
   ```
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

Step 6 Run the following command to perform compilation:

```
make -j
```

Step 7 Run the following command to copy the generated static library to another directory:

```
cp *.a/path/to/EXTRA/mathlib
```

---- End

3.11.4.8 Installing ELPA

Procedure

Step 1 Use PuTTY to log in to the server as the `root` user.

Step 2 Run the following command to decompress the ScaLAPACK installation package:

```
tar -xvf elpa-2019.05.001.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd elpa-2019.05.001
```

Step 4 Run the following command to perform configuration:

```
./configure --prefix=/path/to/EXTRA/elpa --enable-openmp --enable-shared=no
LIBS="/path/to/EXTRA/mathlib/libscalapack.a /path/to/EXTRA/mathlib/liblapack.a /path/to/EXTRA/mathlib/librefblas.a" --disable-sse --disable-sse-assembly --disable-avx --disable-avx2
```

Step 5 Run the following commands to perform compilation:

```
make
```
3.11.4.9 Installing Spglib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the Spglib installation package:

```
tar -xvf spglib-1.12.2.tar.gz
```

Step 3 Run the following commands to switch to the directory generated after decompression:

```
cd spglib-1.12.2
mkdir build
cd build
```

Step 4 Run the following command to perform configuration:

```
cmake .. -DCMAKE_INSTALL_PREFIX="/path/to/EXTRA/spglib112"
```

Step 5 Run the following commands to perform compilation:

```
make -j
make install
```

3.11.4.10 Installing Libxc

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the Libxc installation package:

```
tar -xvf libxc-4.3.4.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd libxc-4.3.4
```

Step 4 Run the following command to perform configuration:

```
./configure FC=gfortran CC=gcc --prefix=/path/to/EXTRA/libxc434
```

Step 5 Run the following commands to perform compilation:

```
make
make install
```

----End
3.11.4.11 Installing GSL

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the GSL installation package:

```bash
tar -xvf gsl-2.6.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```bash
cd gsl-2.6
```

Step 4 Run the following command to perform configuration:

```bash
./configure --prefix=/path/to/EXTRA/gsl
```

Step 5 Run the following commands to perform compilation:

```bash
make
make install
```

Step 6 Run the following commands to set environment variables:

```bash
export LIBRARY_PATH=/path/to/EXTRA/gsl/lib:
export LD_LIBRARY_PATH=/path/to/EXTRA/gsl/lib:
```

----End

3.11.4.12 Installing PLUMED

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the PLUMED installation package:

```bash
tar -xvf plumed-2.5.2.tgz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```bash
cd plumed-2.5.2
```

Step 4 Run the following command to perform configuration:

```bash
./configure CXX=mpicxx CC=mpicc FC=mpifort --prefix=/path/to/EXTRA/plumed252 --enable-external-blas --enable-gsl --enable-external-lapack
LDFLAGS=-L/path/to/EXTRA/mathlib LIBS="-lrefblas -llapack"
```

Step 5 Run the following commands to perform compilation:

```bash
make
make install
```

----End
3.11.5 Obtaining the Source Code

Procedure

**Step 1** Download the CP2K installation package `cp2k-7.1.tar.gz`.
URL: [https://github.com/cp2k/cp2k/archive/v7.1.0.tar.gz](https://github.com/cp2k/cp2k/archive/v7.1.0.tar.gz)

**Step 2** Use the SFTP tool to upload the CP2K installation package to the `/path/to/CP2K` directory on the server.

---End

3.11.6 Compiling and Installing CP2K

Procedure

**Step 1** Use PuTTY to log in to the server as the `root` user.

**Step 2** Run the following command to decompress the CP2K installation package:
```
tar xvf cp2k-7.1.0.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:
```
cd cp2k-7.1.0/arch
```

**Step 4** Run the following command to modify the configuration file:

1. `vi Linux-x86-64-gfortran.psmmp`
2. Press `i` to enter the editing mode and modify the file.
   ```
   GNU_PATH   = /path/to/EXTRA/
   MATHLIBPATH = /path/to/EXTRA/mathlib
   include     $(GNU_PATH)/plumed252/lib/plumed/src/lib/Plumed.inc.static
   ELPA_VER    = 2019.05.001
   ELPA_INC    = $(GNU_PATH)/elpa/include/elpa_openmp-$(ELPA_VER)
   ELPA_LIB    = $(GNU_PATH)/elpa/lib
   FFTW_INC    = $(GNU_PATH)/fftw3/include
   FFTW_LIB    = $(GNU_PATH)/fftw3/lib
  LIBINT_INC   = $(GNU_PATH)/libint2/include
  LIBINT_LIB   = $(GNU_PATH)/libint2/lib
   LIBXC_INC   = $(GNU_PATH)/libxc434/include
   LIBXC_LIB   = $(GNU_PATH)/libxc434/lib
   SPGLIB_INC  = $(GNU_PATH)/spglib112/include
   SPGLIB_LIB  = $(GNU_PATH)/spglib112/lib
   CFLAGS      = -O2 -g -mtune=native
   DFLAGS      = -D_ELPA -D_FFTW3 -D_LIBINT -D_LIBXC
   DFLAGS      += -D_MPI_VERSION=3 -D_PLUMED2 -D_SPGLIB
   DFLAGS      += -D_parallel -D_SCALAPACK
   FCFLAGS     = $(CFLAGS) $(DFLAGS)
   FCFLAGS     += -ffree-form -ffree-line-length-none
   FCFLAGS     += -fopenmp
   FCFLAGS     += -fopen-mp -funroll-loops -std=f2008
   FCFLAGS     += -I$(ELPA_INC)/elpa -I$(ELPA_INC)/modules
   FCFLAGS     += -I$(FTFW_INC) -I$(LIBINT_INC) -I$(LIBXC_INC)
   FCFLAGS     += -I$(PLUMED_DEPENDENCIES) -I$(libelpa_openmp.a
   FCFLAGS     += -I$(LIBELA_LIB)/libelpa_openmp.a
   FCFLAGS     += -I$(LIBXC_LIB)/libxcf03.a
   FCFLAGS     += -I$(LIBINT_LIB)/libint.a
   FCFLAGS     += -I$(LIBINT_LIB)/libint2.a
   FCFLAGS     += -I$(SPGLIB_LIB)/libspglib.a
   LDFLAGS     = $(FCFLAGS)
   LIBS        = $(PLUMED_DEPENDENCIES) -lgs -lplibcblas -lz
   LIBS        += $(ELPA_INC)/libelpa_openmp.a
   LIBS        += $(LIBELA_LIB)/libelpa_openmp.a
   LIBS        += $(LIBXC_LIB)/libxcf03.a
   LIBS        += $(LIBINT_LIB)/libint.a
   LIBS        += $(LIBINT_LIB)/libint2.a
   LIBS        += $(SPGLIB_LIB)/libspglib.a
   ```
Step 5 Run the following commands to go to the directory:

```
  cd /path/to/CP2K/exts/dbcsr
  cp dbcsr-2.1.0-rc12.tar.gz ./
```

Step 6 Run the following commands to go to the directory:

```
  tar -xvf dbcsr-2.1.0-rc12.tar.gz
  cd dbcsr-2.1.0-rc12
```

Step 7 Run the following commands to move all the files in `dbcsr-2.1.0-rc12` to `dbcsr`:

```
  mv * ../
```

Step 8 Run the following command to compile and install CP2K:

```
  make -j 16 ARCH=Linux-x86-64-gfortran VERSION=psmp
```

----End

### 3.11.7 Running and Verifying CP2K

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to load the environment variable:

```
  export PATH=/path/to/CP2K/exe/Linux-x86-64-gfortran:$PATH
```

**Step 3** Run the following command as the root user:

```
  mpirun --allow-run-as-root -np 128 -x OMP_NUM_THREADS=1 cp2k.psmp
  H2O-256.inp > cp2k.H2O-256.inp.log
```

The time difference between Figure 3-8 and Figure 3-9 is 927s.

**Figure 3-8** Output example (1)
3.11.8 More Information

CP2K installation guide:

https://www.cp2k.org/howto:compile

3.12 NWChem 6.8.1 Porting Guide (CentOS 7.6)

3.12.1 Introduction

NWChem is computational chemistry software running on high-performance parallel supercomputers and common workstation clusters. NWChem uses standard quantum mechanics to describe electron wave functions or densities, to calculate the properties of molecules and periodic systems, and to simulate classical molecular dynamics and free energy.

For more information about NWChem, visit the official NWChem website.

Language: Python

Brief description: scalable computational chemistry tools

Open-source protocol: ECL 2.0

Recommended Version

The recommended version is NWChem-6.8.1.

3.12.2 Environment Requirements

Hardware Requirements

Table 3-57 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
Software Requirements

*Table 3-58* lists the software requirements.

**Table 3-58** Software Requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlobalArray</td>
<td>5.6.5</td>
<td><a href="https://github.com/edoapra/ga/releases/download/v5.6.5/ga-5.6.5.tar.gz">https://github.com/edoapra/ga/releases/download/v5.6.5/ga-5.6.5.tar.gz</a></td>
</tr>
<tr>
<td>Test case</td>
<td>siosi8.nw</td>
<td>Test case provided by the software</td>
</tr>
</tbody>
</table>

OS Requirements

*Table 3-59* lists the OS requirements.

**Table 3-59** OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

3.12.3 Planning the Paths for Software Porting

*Table 3-60* lists the software installation paths involved in the NWChem software porting.

**Table 3-60** Paths for Software Porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/NWCHEM</td>
<td>Installation path of NWChem</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

### 3.12.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.

**Procedure**

**Table 3-61 Procedure**

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
</tbody>
</table>

### 3.12.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the NWChem-6.8.1-release.tar.gz installation package.


**Step 2** Use SFTP to upload the NWChem installation package to the /path/to/NWCHEM directory on the server.

----End

### 3.12.6 Compiling and Installing NWChem

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.
Step 2 Run the following commands to set environment variables:

- `export CC=`which gcc`
- `export CXX=`which g++`
- `export FC=`which gfortran`

Step 3 Run the following command to decompress the NWChem installation package:

- `cd /path/to/NWCHEM`
- `tar -zxvf NWChem-6.8.1-release.tar.gz`

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

- `cd NWChem-6.8.1-release`

Step 5 Run the following commands to decompress the GlobalArray software package to the `src/tools` directory:

- `cd src/tools`
- `cp ga-5.6.5.tar.gz /path/to/NWCHEM/NWChem-6.8.1-release/src/tools ./`
- `tar -zxvf ga-5.6.5.tar.gz`

Step 6 Run the following command to perform configuration:

- `export NWCHEM_TOP=/path/to/NWCHEM/NWChem-6.8.1-release`
- `export NWCHEM_TARGET=LINUX64`
- `export NWCHEM_MODULES=all`
- `export USE_MPI=y`
- `export USE_INTERNALBLAS=y`

Step 7 Run the following commands to perform compilation and installation:

- `cd /path/to/NWCHEM/NWChem-6.8.1-release/src`
- `make nwchem_config`
- `make FC=gfortran >& make.log`

Step 8 Run the following command to add the environment variables:

- `export PATH=/path/to/NWCHEM/NWChem-6.8.1-release/bin/LINUX64:SPATH`

--- End

### 3.12.7 Running and Verifying BWA

**Procedure**

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the case folder:

- `cd /path/to/NWCHEM/NWChem-6.8.1-release/web/benchmarks/dft`
Step 3  Run the following command to create the hostfile file:

```
vi hostfile
```

1. Press **i** to enter the editing mode.

```
Node1
Node2
```

2. Press **Esc**, enter :wq!, and press **Enter** to save the settings and exit.

Step 4  Run the following command to clear the cache:

```
clush -w Node[1-2] "sync;echo 3 > /proc/sys/vm/drop_caches"
```

Step 5  Run the following command to perform calculation:

```
mpirun --allow-run-as-root -N 96 -x PATH=$PATH -x LD_LIBRARY_PATH=SLD_LIBRARY_PATH -mca btl ^openib --hostfile hostfile nwchem siosi8.nw -v 2>&1 | tee NWChem_examples.log
```

**NOTE**

- **--hostfile hostfile**: specifies the list of node names to be used. In this example, two nodes are used.
- **-N 96**: indicates that each node runs 96 processes. (The 96-core CPU of a single node is used as an example.)

Step 6  Check the value of **Total times cpu** in **NWChem_examples.log**. The unit is second. A smaller value indicates better performance.

The following is an example of the output result.

**Figure 3-10 Result example**

![](image)

Total times cpu: 43622.7s wall: 43948.7s

----End
3.12.8 Troubleshooting

Problem 1: An Error Is Reported When make NWChem_config Is Run

Symptom:
The error message "guess-mpidefs: command not found..." was displayed when `make NWChem_config` was run.

Possible causes:
The GlobalArray software package is not decompressed to the `src/tools` directory, and the device cannot access the Internet.

Procedure:
Decompress the GlobalArray software package to the `src/tools` directory or connect the device to the Internet.

3.12.9 More Information

Official NWChem website:
https://nwchemgit.github.io/

3.13 ROOT 6.20 Porting Guide (CentOS 7.6)

3.13.1 Introduction

ROOT is a modular scientific software toolkit. It provides all the functions required for big data processing, statistical analysis, visualization, and storage.

For details about ROOT, visit the official ROOT website.

Programming language: C++/Python

Brief description: Modular scientific software toolkit

Open-source license: GPL 3.0

Recommended Version

ROOT 6.20.

3.13.2 Environment Requirements

Hardware Requirements

Table 3-62 lists the hardware requirements.
Table 3-62 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-63 lists the software requirements.

Table 3-63 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT</td>
<td>6.20</td>
<td><a href="https://root.cern.ch/content/release-62000">https://root.cern.ch/content/release-62000</a></td>
</tr>
<tr>
<td>CMake</td>
<td>3.17</td>
<td><a href="https://cmake.org/download/">https://cmake.org/download/</a></td>
</tr>
<tr>
<td>zstd</td>
<td>1.3.4</td>
<td><a href="https://github.com/facebook/zstd/releases/tag/v1.3.4">https://github.com/facebook/zstd/releases/tag/v1.3.4</a></td>
</tr>
<tr>
<td>Test case</td>
<td>benchmarks.C</td>
<td>Test case provided by the software.</td>
</tr>
<tr>
<td></td>
<td>simple</td>
<td>Test case provided by the software.</td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-64 lists the OS requirements.

Table 3-64 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

3.13.3 Planning the Paths for Software Porting

Table 3-65 lists the software installation paths involved in the ROOT software porting.
Table 3-65 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/ZSTD</td>
<td>Installation path of zstd</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/ROOT</td>
<td>Installation path of ROOT</td>
<td></td>
</tr>
</tbody>
</table>

3.13.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Procedure

Table 3-66 Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Installing CMake</td>
<td>For details, see 3.13.4.1 Installing CMake.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Installing zstd</td>
<td>For details, see 3.13.4.2 Installing zstd.</td>
<td></td>
</tr>
</tbody>
</table>

3.13.4.1 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following commands to decompress the CMake installation package:
```
tar -xzvf cmake-3.17.0-rc2.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:
```
cd cmake-3.17.0-rc2
```

Step 4 Run the following commands to perform compilation and installation:
```
./configure --prefix=/path/to/CMAKE
make -j16
make install
```

Step 5 Run the following command to set the CMake environment variable:
```
export PATH=/path/to/CMAKE/bin:$PATH
```

-----End

3.13.4.2 Installing zstd

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the zstd installation package:
```
tar -zxvf zstd-1.3.4.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:
```
cd zstd-1.3.4
```

Step 4 Run the following commands to perform compilation and installation:
```
make -j16
make install
```

Step 5 Run the following commands to set the zstd environment variables:
```
export LD_LIBRARY_PATH=/path/to/ZSTD/lib:$LD_LIBRARY_PATH
```

-----End

3.13.5 Obtaining the Source Code

Procedure

Step 1 Download the root installation package `root_v6.20.00.source.tar.gz`.

Download address: [https://root.cern.ch/content/release-62000](https://root.cern.ch/content/release-62000)
**3.13.6 Compiling and Installing NEMO**

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following commands to decompress the ROOT installation package:

```bash
tar -xzvf root_v6.20.00.source.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd root-6.20.00
```

**Step 4** Run the following command to create and access the directory for compiling the ROOT software:

```bash
mkdir obj
cd obj
```

**Step 5** Ensure that the server is connected to the Internet and run the following commands to compile and install the software:

```bash
cmake ../ -DCMAKE_INSTALL_PREFIX=/opt/xrootd -DENABLE_PERL=FALSE
make -j16
make install
```

**Step 6** Run the following command to add the environment variables:

```bash
source ./bin/thisroot.sh
```

---End

**3.13.7 Running and Verifying the ROOT**

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to switch to the directory where the test case is stored:

```bash
cd /path/to/ROOT/root-6.20.00/obj/tutorials/legacy
```

**Step 3** Run the following command to open ROOT:

```
root
```

**Step 4** Run the following command to perform a benchmark test:

```
.x benchmarks.C
```
Check the value of `ROOTMARKS` in the command output. A larger value indicates better single-core performance.

**Figure 3-11** shows the command output.

**Figure 3-11 Result example**

```
-- ROOT 6.20/00 benchmarks summary (in ROOTMARKS) --
For comparison, a Pentium IV 2.4Ghz is benchmarked at 600 ROOTMARKS
hist1 = 390.34 RealMARKS = 1632.09 CpuMARKS
hsun = 419.53 RealMARKS = 847.06 CpuMARKS
fillrandom = 1737.96 RealMARKS = 600.00 CpuMARKS
tornado = 3305.81 RealMARKS = 180.00 CpuMARKS
na40 = 67238.56 RealMARKS = 83400.00 CpuMARKS
gaus = 163.80 RealMARKS = 3600.00 CpuMARKS
na40jew = 1345.04 RealMARKS = 3600.00 CpuMARKS
ntuple1 = 1432.92 RealMARKS = 840.00 CpuMARKS

* Your machine is estimated at 1010.95 ROOTMARKS *
```

**Step 5** Run the following command to exit ROOT.

```
.q
```

**Step 6** Run the following command to go to the `proof` directory:

```
cd /path/to/ROOT/root-6.20.00/obj/tutorials/proof
```

**Step 7** Run the following command to open ROOT:

```
root
```

**Step 8** Run the following command to perform a simple test:

```
.L runProof.C
runProof("simple(nevt=10000000000000000,nhist=100)","lite://")
```

The test progress is displayed. When the progress reaches 100%, check the value of **Processing rate**. A larger value indicates better multi-core performance.

**Figure 3-12** shows an example of the dialog box.
3.13.8 Troubleshooting

Problem 1: An Error Is Reported When CMake Is Run

**Symptom:**
"Error in <TInterpreter::InspectMembers>: TClass and cling disagree on the size of the class TBaseClass" is displayed when CMake is run.

**Possible Causes:**
The GNU version is incorrect.

**Procedure:**
Check whether the GNU version matches the compilation environment and whether the /usr/local/lib library is correct. For details, see Step 4 and Step 4.

Problem 2: An Error is Reported When the runProof Is Run

**Symptom:**
"SysError in <TUnixSystem::UnixTcpConnect>: connect (localhost:40000) (Connection refused)" is displayed when runProof(simple (nevt = 10000000000000000,nhist=100)) is run.

**Possible Causes:**
You do not need to start the xproofd daemon when runProof is run on a local server.

**Procedure:**
Add the lite:// parameter when running runProof. For details, see Step 8.
3.13.9 More Information

Official ROOT installation guide:

https://root.cern.ch/build-prerequisites#ubuntu

Official ROOT forum:

https://root-forum.cern.ch/

3.14 Geant4 10.6 Porting Guide (CentOS 7.6)

3.14.1 Introduction

Geant4 is a platform for the simulation of the passage of particles through matter using Monte Carlo methods. It is used in high energy physics and nuclear experiments, medical, accelerator and space physics studies.

For more information about Geant4, visit the official Geant4 website.

Programming language: C++

Brief description: Monte Carlo application package

Open-source license: GPL 3.0

Recommended Version

Geant4 10.6

3.14.2 Environment Requirements

Hardware Requirements

Table 3-67 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-68 lists the software requirements.
Table 3-68 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data files</td>
<td>10.6</td>
<td><a href="http://geant4.web.cern.ch/support/download">http://geant4.web.cern.ch/support/download</a></td>
</tr>
<tr>
<td>cmake</td>
<td>3.17</td>
<td><a href="https://cmake.org/download/">https://cmake.org/download/</a></td>
</tr>
<tr>
<td>Test computing instance</td>
<td>exMPI01</td>
<td>Provided by the software.</td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-69 lists the OS requirements.

Table 3-69 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

3.14.3 Planning the Paths for Software Porting

Table 3-70 lists the software installation paths involved in the Geant4 software porting.

Table 3-70 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/GEANT4</td>
<td>Installation path of Geant4</td>
<td></td>
</tr>
</tbody>
</table>

Kunpeng BoostKit for HPC Porting Guide 3 Government HPC

Issue 14 (2021-07-23) Copyright © Huawei Technologies Co., Ltd. 323
<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>/path/to/DATA</td>
<td>Installation path of data file packages</td>
<td>Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

### 3.14.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>For details, see section “Setting Up the Environment for the Cluster Scenario” in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Install CMake.</td>
<td>For details, see 3.14.4.1 Installing CMake.</td>
</tr>
</tbody>
</table>

### 3.14.4.1 Installing CMake

#### Procedure

**Step 1** Use PuTTY to log in to the server as the *root* user.

**Step 2** Decompress the CMake installation package.

```
tar -xzvf cmake-3.17.0-rc2.tar.gz
```

**Step 3** Switch to the directory generated after the package is decompressed.

```
cd cmake-3.17.0-rc2
```

**Step 4** Configure CMake.

```
./configure --prefix=/path/to/CMAKE
```

**Step 5** Compile and install CMake.

```
make -j16
make install
```
Step 6 Set CMake environment variables.

```sh
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

### 3.14.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the Geant4 source code package `geant4.10.06.p01.tar.gz`.


**Step 2** Download 12 data file packages.


**Step 3** Use the SFTP tool to upload the Genet4 source code package to the `/path/to/GEANT4` directory on the server and upload the 12 data file packages to the `/path/to/DATA` directory on the server.

----End

### 3.14.6 Compiling and Installing Geant4

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Configure OpenGL.

```sh
yum install -y mesa*  
yum install -y freeglut*  
yum install -y *GLEW*  
yum install -y openssl-devel
```

**Step 3** Switch to the DATA directory.

```sh
cd /path/to/DATA
```

**Step 4** Create a script for decompression.

1. Run `vi run.sh`.
2. Press i to enter the insert mode and add the following content to decompress data packages in batches:

```bash
#!/bin/bash
for wav in ./*.tar.gz; do
  echo "Extracting wav from $wav"
  tar -xvf $wav
done
```

3. Press Esc, enter `:wq!`, and press Enter to save the file and exit.

**Step 5** Add execution permissions for the script.

```sh
chmod +x run.sh
```
**Step 6** Decompress the data files.

```bash
sh run.sh
```

**Step 7** Set environment variables for data files.

1. **vi data_source**
2. Press `i` to enter the insert mode and add the following content to set environment variables:
   ```bash
   export G4SAIDXSDATA=/path/to/DATA/G4SAIDDATA2.0
   export G4LEDATA=/path/to/DATA/G4EMLOW7.9.1
   export G4LEVELGAMMADATA=/path/to/DATA/PhotonEvaporation5.5
   export G4INCLUDATA=/path/to/DATA/G4INCL1.0
   export G4PARTICLEXSDATA=/path/to/DATA/G4PARTICLEXS2.1
   export G4PIIDATA=/path/to/DATA/G4PII1.3
   export G4RADIOACTIVEDATA=/path/to/DATA/RadioactiveDecay5.4
   export G4REALSURFACEDATA=/path/to/DATA/RealSurface2.1.1
   export G4ABLADATA=/path/to/DATA/G4ABL2.3
   export G4NEUTRONHPDATA=/path/to/DATA/G4NDL4.6
   export G4ENSDFSTATEDATA=/path/to/DATA/G4ENSDFSTATE2.2
   export G4TENDLDATA=/path/to/DATA/G4TENDL1.3.2
   ```
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

**Step 8** Run the following command to make the environment variables take effect:

```bash
source data_source
```

**Step 9** Decompress the Geant4 installation package.

```bash
tar -xvf geant4.10.06.p01.tar.gz
```

**Step 10** Create a `build` folder and switch to the folder.

```bash
mkdir geant4.10.06.p01/build
cd geant4.10.06.p01/build
```

**Step 11** Compile Geant4.

```bash
CC=mpicc CXX=mpicxx CFLAGS='-O3 -finline-functions -march=armv8.2-a -Dflto' cmake -DCMAKE_INSTALL_PREFIX=/path/to/GEANT4/geant4.10.06.p01 -DGEANT4_USE_OPENGL_X11=ON -DGEANT4_BUILD_MULTITHREADED=ON ..
```

**Step 12** Run the following commands to install Geant4:

```bash
make -j16
make install
```

**Step 13** Switch to the parallel module directory.

```bash
cd /path/to/GEANT4/geant4.10.06.p01/examples/extended/parallel/MPI/source
```

**Step 14** Create a `build` folder and switch to the folder.

```bash
mkdir build
cd build
```

**Step 15** Compile the parallel module.


3.14.7 Running and Verifying Geant4

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to switch to the directory where the test computing instance is stored:

```
cd /path/to/GEANT4/geant4.10.06.p01/examples/extended/parallel/MPI/examples
```

**Step 3** Copy all the content in the exMPI01 folder to the exMPI01_bak folder:

```
cp -r exMPI01 exMPI01_bak
```

**Step 4** Switch to the test directory.

```
cd exMPI01
mkdie build
cd build
```

**Step 5** Run the following commands to perform compilation and installation:

```
CC=mpicc CXX=mpicxx CFLAGS='-O3 -finline-functions -march=armv8.2-a -flto' cmake -DCMAKE_INSTALL_PREFIX=/path/to/GEANT4/geant4.10.06.p01-install/lib64 ../
make -j16
```

**Step 6** Run the following command to perform the test:

```
mpirun--allow-run-as-root -np 96 -mca btl ^openib ./exMPI01 run.mac > geant4-128core.log 2>&1
```

In the `geant4-128core.log` file, check the value of `time` (in seconds). A smaller value indicates higher performance.

*Figure 3-13* shows an example of the test result.
3.14.8 Troubleshooting

An Error Reported During the Compilation of the Geant4 Parallel Module

Symptom

When the geant4 parallel module was compiled, "G4MPIManager.hh:117:9: error: 'MPI' does not name a type; did you mean 'M_PI'?" is reported.

Possible Causes

By default, g++ binding is disabled for OpenMPI 4.0.1.

Procedure

Recompile OpenMPI 4.0.1 and add the --enable-mpi-cxx parameter when configuring OpenMPI. For details, see Step 5 in "Installing OpenMPI" in the HPC Solution Basic Environment Setup Guide.

3.14.9 More Information

Official Geant4 website:

http://geant4-data.web.cern.ch

3.15 Ont-Tombo 1.5.1 Porting Guide (CentOS 7.6)

3.15.1 Introduction

Ont-Tombo is a suite of tools primarily for the identification of modified nucleotides from nanopore sequencing data. Ont-Tombo also provides tools for the analysis and visualization of raw nanopore signal.

Ont-Tombo provides the following functions:

- Improved gauze detection
- Visualization of the original reference anchoring signal
- Python API for original model analysis
• User-friendly model estimation methods with tutorials
  For more information about Ont-Tombo, visit the official Hive website.
  Programing language: Python
  Brief description: the tool for the identification of modified nucleotides from nanopore sequencing data
  Open-source protocol: Mozilla Public License Version 2.0

Recommended Version
Ont-tombo 1.5.1

3.15.2 Environment Requirements

Hardware Requirements
Table 3-72 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements
Table 3-73 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDF5</td>
<td>1.10.1</td>
<td><a href="http://hdfgroup.org/HDF5/">http://hdfgroup.org/HDF5/</a></td>
</tr>
<tr>
<td>OpenBLAS</td>
<td>0.3.7</td>
<td><a href="https://github.com/xianyi/OpenBLAS/releases">https://github.com/xianyi/OpenBLAS/releases</a></td>
</tr>
<tr>
<td>Python</td>
<td>2.7.16</td>
<td><a href="https://www.python.org/downloads/release/python-2716/">https://www.python.org/downloads/release/python-2716/</a></td>
</tr>
</tbody>
</table>

OS Requirements
Table 3-74 lists the OS requirements.
Table 3-74 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
</tbody>
</table>

3.15.3 Planning the Paths for Software Porting

Table Paths for software porting lists the software installation paths involved in the Ont-Tombo software porting.

Table 3-75 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS 0.3.7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/PYTHON</td>
<td>Installation path of Python 2.7.16</td>
<td></td>
</tr>
</tbody>
</table>

3.15.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install dependencies.</td>
<td>For details, see 3.15.4.1 Installing Dependencies.</td>
</tr>
<tr>
<td>No.</td>
<td>Operation</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>Install Python.</td>
<td>For details, see 3.15.4.2 Installing Python.</td>
</tr>
<tr>
<td>4</td>
<td>Install PIP.</td>
<td>For details, see 3.15.4.3 Installing PIP.</td>
</tr>
<tr>
<td>5</td>
<td>Install OpenBLAS.</td>
<td>For details, see 3.15.4.4 Installing OpenBLAS.</td>
</tr>
<tr>
<td>6</td>
<td>Install HDF5.</td>
<td>For details, see 3.15.4.5 Installing HDF5.</td>
</tr>
</tbody>
</table>

### 3.15.4.1 Installing Dependencies

**Procedure**

**Step 1** Use PuTTY to log in to the server as the *root* user.

**Step 2** Run the following command to install the dependency:

```
yum install -y zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel
```

---End

### 3.15.4.2 Installing Python

**Procedure**

**Step 1** Use PuTTY to log in to the server as the *root* user.

**Step 2** Run the following commands to decompress the Python installation package:

```
tar zxvf Python-2.7.16.tgz
```

**Step 3** Run the following command to switch to the directory generated after the package is decompressed:

```
cd Python-2.7.16
```

**Step 4** Run the following command to perform configuration:

```
CXX=g++ ./configure --prefix=/path/to/PYTHON
```

**Step 5** Run the following command to perform compilation:

```
make
make install
```

**Step 6** Run the following commands to set the Python environment variables:

```
export PATH=/path/to/PYTHON/bin:SPATH
export LD_LIBRARY_PATH=/path/to/PYTHON/lib:SLD_LIBRARY_PATH
```

**Step 7** Run the following command to check whether the installation is successful:

```
python -V
```
If information similar to the following is displayed, the installation is successful:

```
Python 2.7.16
```

----End

### 3.15.4.3 Installing PIP

**Procedure**

**Step 1** Run the following command to download the distribution file of PIP:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

**Step 2** Run the following command to install PIP.

```
python get-pip.py
```

**Step 3** Run the following command to check whether the installation is successful:

```
which pip
```

If information similar to the following is displayed, the installation is successful:

```
/path/to/PYTHON/bin/pip
```

----End

### 3.15.4.4 Installing OpenBLAS

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the OpenBLAS installation package:

```
tar -xvf OpenBLAS-0.3.7.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after the package is decompressed:

```
cd OpenBLAS-0.3.7
```

**Step 4** (Optional) If the GCC compiler version is not GNU 9 or later, modify the Makefile.arm64 file and run the gcc -v command to query the GCC compiler version.

1. `vi Makefile.arm64`
2. Press `I` to enter the editing mode and delete the content in the red box shown in the following figure.

```
ifeq ($(CORE), TSV110)
COMMON_OPT += -march=armv8.2-a -mtune=tsv110
FCOMMON_OPT += -march=armv8.2-a -mtune=tsv110
endif
```

3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.
Step 5 Run the following commands to install OpenBLAS.

```
make BINARY=64 FC=/usr/bin/gfortran USE_THREAD=1
make PREFIX=/path/to/OPENBLAS install
```

Step 6 Set the OpenBLAS environment variable.

```
export PATH=/path/to/OPENBLAS/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

3.15.4.5 Installing HDF5

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the HDF5 installation package:

```
tar -xvf hdf5-1.10.1.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd hdf5-1.10.1
```

Step 4 Run the following command to perform configuration:

```
FC=/usr/bin/gfortran ./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-shared
```

Step 5 Run the following commands to perform compile and install HDF5:

```
make -j 16
make install
```

Step 6 Run the following commands to set the HDF5 environment variables:

```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
```

3.15.5 Obtaining the Source Code

Procedure

Step 1 The Ont-Tombo software package is downloaded and installed through the link specified by the pip command. You do not need to download the source code package separately.

URL: https://pypi.tuna.tsinghua.edu.cn/simple

-----End
3.15.6 Compiling and Installing Ont-Tombo

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install Numpy:

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple numpy
```

Step 3 Run the following commands to install Scipy.

```
LAPACK=/path/to/OPENBLAS/lib/libopenblas.so BLAS=/path/to/OPENBLAS/lib/libopenblas.so pip install -i https://pypi.tuna.tsinghua.edu.cn/simple scipy
```

Step 4 Run the following command to install Ont-Tombo:

```
CFLAGS="-I/path/to/HDF5/include -L/path/to/HDF5/lib" pip install -i https://pypi.tuna.tsinghua.edu.cn/simple ont-tombo
```

If information similar to the following is displayed, the installation is successful:

```
Successfully installed cython-0.29.15 future-0.18.2 h5py-2.10.0 mappy-2.17 ont-tombo-1.5.1 tqdm-4.43.0
```

----End

3.16 DL_POLY 1.10 Porting Guide (CentOS 7.6)

3.16.1 Introduction

DL_POLY is a general purpose classical molecular dynamics (MD) simulation software developed at Daresbury Laboratory by I.T. Todorov and W. Smith.

For details about DL_POLY, visit the [official DL_POLY website](https://www.daresbury.ac.uk/software/dl-poly/).

Recommended Version

DL_POLY 1.10

3.16.2 Environment Requirements

Hardware Requirements

Table 3-76 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-77 lists the software requirements.
### Table 3-77 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL_POLY</td>
<td>V1.10</td>
<td><a href="https://gitlab.com/DL_POLY_Classic/dl_poly/-/archive/RELEASE-1-10/dl_poly-RELEASE-1-10.tar.gz">https://gitlab.com/DL_POLY_Classic/dl_poly/-/archive/RELEASE-1-10/dl_poly-RELEASE-1-10.tar.gz</a></td>
</tr>
<tr>
<td>Test case</td>
<td>TEST41.tar.gz</td>
<td><a href="https://gitlab.com/DL_POLY_Classic/tests/-/blob/master/TEST41.tar.gz">https://gitlab.com/DL_POLY_Classic/tests/-/blob/master/TEST41.tar.gz</a></td>
</tr>
</tbody>
</table>

### OS Requirements

Table 3-78 lists the OS requirements.

### Table 3-78 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 3.16.3 Planning the Paths for Software Porting

Table 3-79 describes the software installation paths involved in the DL_POLY software porting.

### Table 3-79 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation PathHPC Solution Basic Environment Setup Guide</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/DL_POLY</td>
<td>Installation path of DL_POLY.</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>
3.16.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 3-80 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the</td>
</tr>
</tbody>
</table>

3.16.5 Obtaining the Source Code

Procedure

**Step 1** Download the DL_POLY installation package `dl_poly-RELEASE-1-10.tar.gz`.
URL: [https://gitlab.com/DL_POLY_Classic/dl_poly/-/archive/RELEASE-1-10/dl_poly-RELEASE-1-10.tar.gz](https://gitlab.com/DL_POLY_Classic/dl_poly/-/archive/RELEASE-1-10/dl_poly-RELEASE-1-10.tar.gz)

**Step 2** Use SFTP to upload the DL_POLY installation package to the `/path/to/DL_POLY` directory on the server.

----End

3.16.6 Compiling and Installing DL_POLY

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the DL_POLY installation package:

```
tar -xzf dl_poly-RELEASE-1-10.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:

```
cd dl_poly-RELEASE-1-10/source
```

**Step 4** Copy the Makefile file.

```
cp ../build/MakePAR ./Makefile
```

**Step 5** Modify the Makefile file.

```
vim Makefile
```

Change the content in line 65 to the following:

```
FC=mpif90 FFLAGS="-c -O3 -march=armv8.2-a -mtune=tsv110" \
```
Step 6  Run the following command to perform compilation and installation:

make dlpoly

Step 7  Run the following command to set the environment variable of DL_POLY:

export PATH=/path/to/DL_POLY/dl_poly-RELEASE-1-10/execute:$PATH

Step 8  Run the following command to check whether DL_POLY is installed successfully:

which DLPOLY.X

If information similar to the following is displayed, the installation is successful:

/path/to/DL_POLY/dl_poly-RELEASE-1-10/execute/DLPOLY.X

----End

3.16.7 Running and Verifying DL_POLY

3.16.7.1 Running and Verifying DL_POLY on a Single Node

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to decompress the TEST41.tar.gz test case package:

tar zxf TEST41.tar.gz

Step 3  Go to the TEST41/NHC directory.

cd TEST41/NHC

Step 4  Run the following command to run the test case in single-node mode:

mpirun --allow-run-as-root -np 128 -mca btl ^openib DLPOLY.X

Step 5  View the execution result.

time elapsed since job start = XX seconds in the OUTPUT file indicates the performance. A smaller value of XX indicates better performance.

Figure 3-14 is an example of the output result.

Figure 3-14 Result example

![Figure 3-14 Result example]

----End

3.16.7.2 Running and Verifying DL_POLY in Multi-Node Mode

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to decompress the TEST41.tar.gz test case package:
tar xzf TEST41.tar.gz

**Step 3** Go to the TEST41/NHC directory.

cd TEST41/NHC

**Step 4** Run the following command to create the **hostfile** file and add node information:

```bash
echo -e 'node1
node2
…
nodex' > /path/to/HOSTFILE
```

**NOTE**

In the preceding command, `node1`, `node2`, ..., and `nodex` indicate the host names of the nodes. You can run the `hostname` command to query the host names. `\n` indicates a newline character.

**Step 5** On `node2`, run the following command to configure environment variables:

```bash
vi /root/.bashrc
```

**Step 6** Press `i` to enter the editing mode, and add the following content to the end of the `.bashrc` file:

```bash
export PATH=/path/to/GNU/bin:/path/to/OPENMPI/bin:SPATH
export LD_LIBRARY_PATH=/path/to/GNU/lib64:/path/to/OPENMPI/lib:
SLD_LIBRARY_PATH
export PATH=/path/to/DL_POLY/dl_poly-RELEASE-1-10/execute:SPATH
```

**Step 7** Run the following command to run the test case in multi-node mode:

```bash
mpirun --allow-run-as-root -np 256 -N 128 -x OMP_NUM_THREADS=1 --hostfile /path/to/HOSTFILE -mca btl ^openib DLPOLY.X
```

**Step 8** View the execution result.

```bash
time elapsed since job start = XX seconds
```

The `time elapsed since job start` indicates the performance. A smaller value of `XX` indicates better performance.

https://www.scd.stfc.ac.uk/Pages/DL_POLY.aspx

**Figure 3-15** is an example of the output result.

![Figure 3-15](image)

**Figure 3-15 Result example**

```bash
6.7138 1.023616E+00 6.380668E+01
6.7633 1.018976E+00 7.042704E+01
6.8136 1.018871E+00 7.269998E+01
6.8639 1.016988E+00 7.373420E+01
6.9141 1.017692E+00 7.542390E+01
6.9644 1.016372E+00 7.713620E+01
7.0147 1.012845E+00 7.880599E+01
7.0650 1.011887E+00 8.062332E+01
7.1153 1.016353E+00 8.240002E+01
7.1656 1.004753E+00 8.419193E+01
7.2159 1.003921E+00 8.600757E+01
```

```bash
time elapsed since job start = 92.120 seconds
```

-----End
3.16.8 More Information

For more information, visit the official DL_POLY website:

https://www.scd.stfc.ac.uk/Pages/DL_POLY.aspx

3.17 Gamess Porting Guide (CentOS 7.6)

3.17.1 Introduction

The General Atomic and Molecular Electronic Structure System (GAMESS) is a program for \textit{ab initio} quantum chemistry. GAMESS can compute SCF wavefunctions ranging from RHF, ROHF, UHF, GVB, and MCSCF. Correlation corrections to these SCF wave functions include Configuration Interaction, second order perturbation Theory, and Coupled-Cluster approaches, as well as the Density Functional Theory approximation. Excited states can be computed by CI, EOM, or TD-DFT procedures. Nuclear gradients are available for automatic geometry optimization, transition state searches, or reaction path following. Computation of the energy hessian permits prediction of vibrational frequencies, with IR or Raman intensities. Solvent effects may be modeled by the discrete Effective Fragment potentials, or continuum models such as the Polarizable Continuum Model. Numerous relativistic computations are available, including infinite order two component scalar relativity corrections, with various spin-orbit coupling options. The Fragment Molecular Orbital method permits use of many of these sophisticated treatments on very large systems, by dividing the computation into small fragments. Nuclear wavefunctions can also be computed in VSCF or with explicit treatment of nuclear orbitals by the NEO code.

For more information, visit the GAMESS official website.

Programming language: C/Fortran 90

Brief description: GAMESS can compute SCF wavefunctions ranging from RHF, ROHF, UHF, GVB, and MCSCF.

Open-source license: GPL 3.0

Recommended Version

GAMESS does not have the version number. It is versioned based on the date of the public release. The version used in this document is R2 released on September 30, 2020.

3.17.2 Environment Requirements

Hardware Requirements

Table 3-81 lists the hardware requirements.
### Table 3-81 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

### Software Requirements

Table 3-82 lists the software requirements.

#### Table 3-82 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAMESS</td>
<td>-</td>
<td><a href="https://www.msg.chem.iastate.edu/GAMESS/download/register">https://www.msg.chem.iastate.edu/GAMESS/download/register</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.1</td>
<td><a href="https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz">https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz</a></td>
</tr>
<tr>
<td>NetCDF-C</td>
<td>4.7.0</td>
<td><a href="https://www.unidata.ucar.edu/downloads/netcdf/">https://www.unidata.ucar.edu/downloads/netcdf/</a></td>
</tr>
<tr>
<td>NetCDF-Fortran</td>
<td>4.4.5</td>
<td><a href="https://www.unidata.ucar.edu/downloads/netcdf/">https://www.unidata.ucar.edu/downloads/netcdf/</a></td>
</tr>
<tr>
<td>Test case</td>
<td>runall</td>
<td>Provided by the software.</td>
</tr>
</tbody>
</table>

### OS Requirements

Table 3-83 lists the OS requirements.

#### Table 3-83 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 3.17.3 Planning the Paths for Software Porting

This section describes the software installation paths involved in the GAMESS software porting.
### Table 3-84 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>. The installation paths listed in this table are only for reference. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HDF5</td>
<td>Installation path of HDF5.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>/path/to/NETCDF</td>
<td>Installation path of NetCDF-C.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/GAMESS</td>
<td>Installation path of GAMESS.</td>
<td></td>
</tr>
</tbody>
</table>

### 3.17.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Configuration Process

### Table 3-85 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Installing HDF5</td>
<td>For details, see <a href="#">3.17.4.1 Installing HDF5</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Install NetCDF-C.</td>
<td>For details, see <a href="#">3.17.4.2 Installing NetCDF-C</a>.</td>
</tr>
<tr>
<td>4</td>
<td>Install NetCDF-fortran.</td>
<td>For details, see <a href="#">3.17.4.3 Installing NetCDF-fortran</a>.</td>
</tr>
</tbody>
</table>
3.17.4.1 Installing HDF5

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the HDF5 installation package:
```
tar -zxvf hdf5-1.10.1.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:
```
cd hdf5-1.10.1
```

**Step 4** Run the following commands to create the installation directory:
```
mkdir -p /path/to/HDF5
```

**Step 5** Run the following command to perform configuration:
```
./configure --prefix=/path/to/HDF5 --build=aarch64-unknown-linux-gnu --enable-fortran --enable-static=yes --enable-parallel --enable-shared CC=mpicc CXX=mpicxx FC=mpifort F77=mpifort
```

**Step 6** Run the following commands to perform compilation and installation:
```
make -j 16
make install
```

**Step 7** Run the following commands to set HDF5 environment variables:
```
export PATH=/path/to/HDF5/bin:$PATH
export LD_LIBRARY_PATH=/path/to/HDF5/lib:$LD_LIBRARY_PATH
export HDF5=/path/to/HDF5
```

----End

3.17.4.2 Installing NetCDF-C

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the NetCDF-C installation package:
```
tar -zxvf netcdf-c-4.7.0.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:
```
cd netcdf-c-4.7.0
```

**Step 4** Run the following command to perform configuration:
```
./configure --prefix=/path/to/NETCDF LDFLAGS="-L$HDF5/lib" CPPFLAGS="-I$HDF5/include" CC=mpicc --disable-dap
```

----End
Step 5 Run the following commands to perform compilation and installation:

```
make -j
make install
```

Step 6 Run the following commands to set NetCDF-C environment variables:
```
export PATH=/path/to/NETCDF/bin:$PATH
export LD_LIBRARY_PATH=/path/to/NETCDF/lib:$LD_LIBRARY_PATH
export NETCDF=/path/to/NETCDF
```

3.17.4.3 Installing NetCDF-fortran

Procedure

Step 1 Use PuTTY to log in to the server as the `root` user.

Step 2 Run the following command to decompress the NetCDF-Fortran installation package:

```
tar -zxvf netcdf-fortran-4.4.5.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:
```
cd netcdf-fortran-4.4.5
```

Step 4 Run the following command to perform configuration:
```
./configure --prefix=/path/to/NETCDF CPPFLAGS="-I$HDF5/include -I$NETCDF/include" LDFLAGS="-L$HDF5/lib -L$NETCDF/lib" CC=mpicc FC=mpif90 F77=mpif90
```

Step 5 Run the following commands to perform compilation and installation:
```
make -j
make install
```

3.17.5 Obtaining the Source Code

Academic and industrial users can obtain GAMESS site licenses free of charge. During the download, the system will ask you to agree to this license. This license recognizes Gordon Group’s exclusive rights to GAMESS and also prohibits you from making copies of the GAMESS code for any purpose. You can only use the code in your own organization.

Procedure

Step 1 Register and apply for a license at

https://www.msg.chem.iastate.edu/GAMESS/download/register/
Step 2 After completing the registration, you will receive an email containing the download password. You can use any browser to download the compressed tar file or precompiled binary file of the source code.

3.17.6 Compiling and Installing GAMESS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Check whether the required software is installed:

```
rpm -qa |grep gcc
```

```
<table>
<thead>
<tr>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc-4.8.5-36.el7.aarch64</td>
</tr>
<tr>
<td>gcc-gfortran-4.8.5-36.el7.aarch64</td>
</tr>
<tr>
<td>libgcc-4.8.5-36.el7.aarch64</td>
</tr>
<tr>
<td>gcc-c++-4.8.5-36.el7.aarch64</td>
</tr>
</tbody>
</table>
```

Step 3 Run the following command to create a main program installation directory:

```
mkdir -p /path/to/GAMESS
```

Step 4 Run the following command to copy the installation package to the main program installation directory:

```
cp gamess-current.tar.gz /path/to/GAMESS
```

Step 5 Run the following command to go to the main program installation directory:

```
cd /path/to/GAMESS
```

Step 6 Run the following commands to decompress the installation package and obtain the root permission:

```
tar -zxvf gamess-current.tar.gz
```
```
chown -R root:root gamess
```

Step 7 Install the GAMESS main program.

1. Run the following commands to go to the GAMESS installation directory:
```
cd /path/to/GAMESS
```
```
./config
```
2. Press Enter and enter linux64.
3. Press Enter three times and enter 00.
4. Press Enter and enter gfortran.
5. Press Enter and enter the gfortran version number obtained in Step 2.
6. Press Enter twice and enter none.
7. Press Enter three times and enter sockets.
8. Press Enter and enter no.
9. Press Enter and enter no.
10. Press Enter and enter no.
11. Press Enter.
   The configuration file install.info is generated in /path/to/GAMESS.

**Step 8** Compile the GAMESS main program.

1. Run the following commands to compile DDI:
   ```
   cd ddi
   ./compddi
   mv ddikick.x ../
   cd ..
   ```

2. Run the following commands to compile GAMESS. After the command is executed for 10 minutes, many configuration files are generated in the object directory.
   ```
   ./compall
   ```

3. Run the following commands to set the GAMESS link to generate games.00.x:
   ```
   ./lked gamess 00
   ```

4. Change the run environment.
   a. Run the following command to create the tmp directory for storing temporary files:
      ```
      mkdir tmp
      ```
   b. Run the following command to open the configuration file:
      ```
      vi rungms
      ```
   c. Enter i and enter:
      ```
      set SCR=/path/to/GAMESS/tmp
      set USERSCR=/path/to/GAMESS/tmp
      set GMSPATH=/path/to/GAMESS
      ```
   d. Press ESC, enter :wq, and press Enter to save the modification and exit.

-----End

### 3.17.7 Running and Verifying GAMESS

**Procedure**

**Step 1** Run the following command to check the host name:
```
hostname
```

**Step 2** Run the following command to change the host name in the runall file:
```
vi runall
```

**Step 3** Change si.msg.chem.iastate.edu in line 23 to hostname.

**Step 4** Run the following command to run the test case:
```
time ./runall 00
```

**Step 5** Move the 47 log files in the /path/to/GAMESS directory to the /path/to/GAMESS/tests/standard directory.

**Step 6** Check whether the test case is executed successfully.
1. Run the following command to switch to the directory for storing the test case:
   ```
   cd tests/standard
   ```
2. Run the following command to modify the `checktst` file:
   ```
   vim checktst
   ```
   Change 48 in line 39 to 47.
3. Run the following command to perform the check:
   ```
   time ./checktst
   ```
   If the following information is displayed, GAMESS is installed successfully:
   ```
   All 47 testresults are correct!
   ```

---End

3.17.8 More Resources

GAMESS official website:

https://www.msg.chem.iastate.edu/GAMESS/

3.18 CPMD 4.1 Porting Guide (CentOS 7.6)

3.18.1 Introduction

CPMD is a parallelized plane wave and pseudopotential implementation of Density Functional Theory, particularly designed for ab-initio molecular dynamics.

For more information about CPMD, visit the CPMD official website.

Programming language: Fortran

Brief description: a piece of molecular dynamics simulation software.

Recommended Software Version

CPMD 4.1

3.18.2 Environment Requirements

Hardware Requirements

Table 3-86 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
Software Requirements

Table 3-87 lists the software requirements.

Table 3-87 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPMD</td>
<td>4.1</td>
<td><a href="https://www.cpmd.org/wordpress/">https://www.cpmd.org/wordpress/</a></td>
</tr>
<tr>
<td>OpenBLAS</td>
<td>0.3.6</td>
<td><a href="https://github.com/xianyi/OpenBLAS/archive/v0.3.6.tar.gz">https://github.com/xianyi/OpenBLAS/archive/v0.3.6.tar.gz</a></td>
</tr>
<tr>
<td>FFTW</td>
<td>3.3.8</td>
<td><a href="http://www.fftw.org/fftw-3.3.8.tar.gz">http://www.fftw.org/fftw-3.3.8.tar.gz</a></td>
</tr>
<tr>
<td>Test cases</td>
<td>04-gly</td>
<td><a href="https://ibm.ent.box.com/v/CPMD-Tutorial-Full-Archive">https://ibm.ent.box.com/v/CPMD-Tutorial-Full-Archive</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-88 lists the OS requirements.

Table 3-88 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

3.18.3 Paths for Software Porting

This chapter describes the software installation paths involved in the CPMD software porting.

Table 3-89 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/path/to/KPGCC</td>
<td>Installation path of the Kunpeng GCC.</td>
<td>For details, see section &quot;Kunpeng GCC Installation and Operation&quot; in User Guide (Kunpeng GCC).</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HMPI</td>
<td>Installation path of the Hyper MPI.</td>
<td>For details, see section &quot;Source Code Installation&quot; in Hyper MPI User Guide.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/SRC</td>
<td>Path of the source package.</td>
<td>The installation paths provided here are only an example. Shared</td>
</tr>
</tbody>
</table>
### 3.18.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

#### Configure Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the open source environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Setting up the Huawei Kunpeng compiler environment</td>
<td>For details, see section &quot;Installing the Kunpeng GCC&quot; in User Guide (Kunpeng GCC).</td>
</tr>
<tr>
<td>3</td>
<td>Setting up the Huawei MPI environment</td>
<td>For details, see section &quot;Installation Introduction&quot; in Hyper MPI User Guide.</td>
</tr>
<tr>
<td>4</td>
<td>Installing OpenBLAS</td>
<td>See 3.18.4.1 Installing OpenBLAS.</td>
</tr>
<tr>
<td>5</td>
<td>Installing FFTW</td>
<td>See 3.18.4.2 Installing FFTW.</td>
</tr>
</tbody>
</table>
3.18.4.1 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the OpenBLAS installation package:

\[
\text{cd } /\text{path/to/SRC} \\
\text{tar -xvf OpenBLAS-0.3.6.tar.gz}
\]

Step 3 Run the following command to go to the directory generated after decompression:

\[
\text{cd OpenBLAS-0.3.6}
\]

Step 4 Run the following commands to perform the compilation and installation:

\[
\text{make} \\
\text{make PREFIX=}
\text{/path/to/OPENBLAS } \text{install}
\]

Step 5 Run the following commands to configure the environment variables:

\[
\text{export PATH=}
\text{/path/to/OPENBLAS/bin:$PATH} \\
\text{export LD_LIBRARY_PATH=}
\text{/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH}
\]

----End

3.18.4.2 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the FFTW installation package:

\[
\text{cd } /\text{path/to/SRC} \\
\text{tar -xvf fftw-3.3.8.tar.gz}
\]

Step 3 Run the following command to go to the directory generated after decompression:

\[
\text{cd fftw-3.3.8}
\]

Step 4 Run the following command to execute the bootstrap.sh script:

\[
./bootstrap.sh
\]

Step 5 Run the following command to perform the configuration:

\[
./\text{configure }\text{--prefix=}
\text{/path/to/FFTW} \text{--enable-shared } \text{--enable-static } \text{--enable-fma } \text{--enable-neon } \text{--enable-mpi } \text{--enable-threads}
\]

Step 6 Run the following commands to perform the compilation and installation:

\[
\text{make -j} \\
\text{make -j install}
\]

----End
3.18.5 Obtaining the Source Code

Procedure

Step 1  Register with the official website and apply for and download the CPMD installation package cpmd-v4.1.tar.gz.

Download address: https://www.cpmd.org/wordpress/index.php/download/

Step 2  Download the test case file 04-gly.

Download address: https://ibm.ent.box.com/v/CPMD-Tutorial-Full-Archive

Step 3  Use an SFTP tool to upload the CPMD installation package to the /path/to/SRC directory on the server, and upload the test case file to the /path/to/CASE directory on the server.

----End

3.18.6 Compilation and Installation

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to decompress the CPMD installation package:

```
cd /path/to/SRC
.tar xvf cpmd-v4.1.tar.gz
```

Step 3  Run the following command to go to the directory generated after decompression:

```
cd CPMD
```

Step 4  Run the following commands to modify the LINUX-GFORTRAN-SLES11-MPI-FFTW file:

1. Open the file.

```
vi configure/LINUX-GFORTRAN-SLES11-MPI-FFTW
```

2. Press i to go to the edit mode.

   - Modify lines 21 and 22:
     
     ```
     FFLAGS='-O2 -ffixed-line-length-none -ffree-form -ffree-line-length-none -march=armv8.2-a -mtune=tsv110'
     CFLAGS='-O3 -Wall'
     ```
   
   - Modify line 26:
     
     ```
     LIBS='-L/path/to/FFTW/lib -lfftw3_threads -L/path/to/OPENBLAS/lib -lopenblas'
     ```
   
   - Modify line 28:
     
     ```
     LIBS='-L/path/to/FFTW/lib -lfftw3 -L/path/to/OPENBLAS/lib -lopenblas'
     ```

3. Press Esc, type :wq!, and press Enter to save the file and exit.

Step 5  Run the following command to perform the configuration:

```
./configure.sh LINUX-GFORTRAN-SLES11-MPI-FFTW
```

Step 6  Run the following command to perform the compilation and installation:

```
make -j
```
Step 7  Run the following command to copy the executable program:

```bash
cp /path/to/SRC/CPMD/bin/cpmd.x /path/to/CPMD
```

Step 8  Run the following commands to view the generated executable file:

```bash
ll /path/to/CPMD
```

```
-rwxr-xr-x 1 root root 6764040 Feb 24 17:54 cpmd.x
```

Step 9  Run the following command to configure the environment variables:

```bash
export PATH=/path/to/CPMD:$PATH
```

-----End

### 3.18.7 Running and Verification

#### Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to go to the test case directory and decompress the test case file:

```bash
cd /path/to/CASE
unzip CPMD_Tutorial_Final_Archive.zip
```

Step 3  Run the following command to go to the test case directory:

```bash
cd CPMD_Tutorial_Final_Archive/inputs/04-gly
```

Step 4  Run the following commands to perform the preprocessing:

```bash
cd 1-gly-opt
mpirun --allow-run-as-root -np 128 cpmd.x ../1-gly-opt.inp ../Files/ >& 1-gly-opt.out
```

Step 5  Run the following commands to execute the test case:

```bash
cd ../2-gly-md-equilib
```

```bash
cat << END > LATEST
../1-gly-opt/RESTART.1
1
END
```

```bash
mpirun --allow-run-as-root -np 128 cpmd.x ../2-gly-md-equilib.inp ../Files/ >& 2-gly-md-equilib.out
```

Step 6  After the execution is complete, check the value of **ELAPSED TIME** in the `2-gly-md-equilib.out` file. A smaller value indicates a higher performance.

-----End
3.18.8 More Resources

CPMD official website:

https://www.cpmd.org/wordpress/

3.19 ESPResSo 4.1.4 Porting Guide (CentOS 7.6)

3.19.1 Introduction

ESPResSo is a highly versatile software package for performing and analyzing scientific Molecular Dynamics many-particle simulations of coarse-grained atomistic or bead-spring models. It can be used to simulate systems such as polymers, liquid crystals, colloids, polyelectrolytes, ferrofluids and biological systems, for example DNA and lipid membranes.

For more information about ESPResSo, visit the ESPResSo official website.

Programing languages: C++/Python

Brief description: a highly versatile software package for performing and analyzing scientific Molecular Dynamics many-particle simulations of coarse-grained atomistic or bead-spring models.

Open-source license: GPL 3.0

Recommended Software Version

ESPResSo-4.1.4

3.19.2 Environment Requirements

Hardware Requirements

Table 3-91 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-92 lists the software requirements.
### Table 3-92 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESPResSo</td>
<td>4.1.4</td>
<td><a href="https://github.com/espressomd/espresso/releases/download/4.1.4/espresso-4.1.4.tar.gz">https://github.com/espressomd/espresso/releases/download/4.1.4/espresso-4.1.4.tar.gz</a></td>
</tr>
<tr>
<td>FFTW</td>
<td>3.3.8</td>
<td><a href="http://www.fftw.org/fftw-3.3.8.tar.gz">http://www.fftw.org/fftw-3.3.8.tar.gz</a></td>
</tr>
<tr>
<td>boost</td>
<td>1.72.0</td>
<td><a href="https://dl.bintray.com/boostorg/release/1.72.0/source/boost_1_72_0.tar.gz">https://dl.bintray.com/boostorg/release/1.72.0/source/boost_1_72_0.tar.gz</a></td>
</tr>
</tbody>
</table>

### OS Requirements

Table 3-93 lists the OS requirements.

### Table 3-93 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 3.19.3 Paths for Software Porting

This chapter describes the software installation paths involved in the ESPResSo software porting.

### Table 3-94 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/path/to/KPGCC</td>
<td>Installation path of the Kunpeng GCC.</td>
<td>For details, see section &quot;Kunpeng GCC Installation and Operation&quot; in <a href="http://www.centos.org/download/">User Guide (Kunpeng GCC)</a></td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HMPI</td>
<td>Installation path of the Hyper MPI.</td>
<td>For details, see section &quot;Source Code Installation&quot; in <a href="http://www.centos.org/download/">Hyper MPI User Guide</a></td>
</tr>
<tr>
<td>3</td>
<td>/path/to/ESPRESSOMD</td>
<td>Installation path of ESPResSo.</td>
<td>The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in <a href="http://www.centos.org/download/">Kunpeng BoostKit for HPC Porting Guide</a></td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>4</td>
<td>/path/to/FFTW</td>
<td>Installation path of FFTW.</td>
<td>this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>5</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/BOOST</td>
<td>Installation path of BOOST.</td>
<td></td>
</tr>
</tbody>
</table>

### 3.19.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

#### Configure Process

**Table 3-95 Configure process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the open source environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Setting up the Huawei Kunpeng compiler environment</td>
<td>For details, see section &quot;Installing the Kunpeng GCC&quot; in <em>User Guide (Kunpeng GCC)</em>.</td>
</tr>
<tr>
<td>3</td>
<td>Setting up the Huawei MPI environment</td>
<td>For details, see section &quot;Installation Introduction&quot; in <em>Hyper MPI User Guide</em>.</td>
</tr>
<tr>
<td>3</td>
<td>Installing dependencies using Yum</td>
<td>See 3.19.4.1 Installing Dependencies Using Yum.</td>
</tr>
<tr>
<td>4</td>
<td>Installing FFTW</td>
<td>See 3.19.4.2 Installing FFTW.</td>
</tr>
<tr>
<td>5</td>
<td>Installing CMake</td>
<td>See 3.19.4.3 Installing CMake.</td>
</tr>
<tr>
<td>6</td>
<td>Installing BOOST</td>
<td>See 3.19.4.4 Installing BOOST.</td>
</tr>
</tbody>
</table>
3.19.4.1 Installing Dependencies Using Yum

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to configure the HUAWEI CLOUD mirror source:

```
wget --no-check-certificate -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.tools.huawei.com/repository/conf/CentOS-AltArch-7.repo
echo '^[epel]
name=Extra Packages for Enterprise Linux 7 - $basearch
b baseURL=http://mirrors.tools.huawei.com/epel/7/$basearch
failovermethod=priority
enabled=1
gpgcheck=0'$ > epel.repo
```

Step 3 Run the following commands to configure the software source cache:

```
yum clean all
yum makecache
```

Step 4 Run the following command to install python3, GSL, git, numpy, and Cython:

```
yum install -y python36 python3-devel gsl-devel git python36-numpy.aarch64 python36-Cython.aarch64
```

----End

3.19.4.2 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the FFTW installation package:

```
tar -xvf fftw-3.3.8.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd fftw-3.3.8
```

Step 4 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon
```

Step 5 Run the following commands to perform the compilation and installation:

```
make -j 128
make install
```
Step 6  Run the following command to configure the FFTW environment variables required for installing the main program:

```
export CPATH=/path/to/FFTW/include:SPATH
```

-----End

3.19.4.3 Installing CMake

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to decompress the CMake installation package:

```
tar -xvf cmake-3.8.1.tar.gz
```

Step 3  Run the following command to go to the directory generated after decompression:

```
cd CMake-3.8.1
```

Step 4  Run the following commands to perform the compilation and installation:

```
./bootstrap --prefix=/path/to/CMAKE
make -j 16
make install
```

Step 5  Run the following command to configure CMake environment variables:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

-----End

3.19.4.4 Installing BOOST

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to decompress the BOOST installation package:

```
tar -xvf boost_1_72_0.tar.gz
```

Step 3  Run the following command to go to the directory generated after decompression:

```
cd boost_1_72_0
```

Step 4  Run the following command to perform the configuration:

```
./bootstrap.sh --with-toolset=gcc --with-libraries=all --with-python=/usr/bin/python3 --with-python-root=/usr --with-python-version=3.6 --prefix=/path/to/BOOST
```

Step 5  Run the following commands to modify the `project-config.jam` file:

1. Open the `project-config.jam` file.

```
vi project-config.jam
```
2. Press `i` to go to the edit mode.
   - Change the semicolon (;) in line 21 to a colon (:) and add two lines after line 21.

   Before the modification:
   ```
   21     using python : 3.6 : "/usr" ;
   ```

   After the modification:
   ```
   21     using python : 3.6 : "/usr" :
   22     "/usr/include/python3.6m" :
   23     "/usr/lib" ;
   ```
   - Create a new line under the last line:

     ```
     using mpi ;
     ```

3. Press `Esc`, type `.wq!`, and press `Enter` to save the file and exit.

   **Step 6** Run the following commands to perform the compilation and installation:
   ```
   ./b2 -j32
   ./b2 install -j32
   ```

   **Step 7** Run the following commands to create a `BOOST_ROOT` variable and set the environment variables:
   ```
   export BOOST_ROOT=/path/to/BOOST
   export LD_LIBRARY_PATH=/path/to/BOOST/lib:$LD_LIBRARY_PATH
   ```

   ----End

3.19.5 Obtaining the Source Code

**Procedure**

   **Step 1** Download the ESPResSo installation package `espresso-4.1.4.tar.gz`.
   Download address: [https://github.com/espressomd/espresso/releases/download/4.1.4/espresso-4.1.4.tar.gz](https://github.com/espressomd/espresso/releases/download/4.1.4/espresso-4.1.4.tar.gz)

   **Step 2** Use an SFTP tool to upload the ESPResSo installation package to the `/path/to/ESPRESSOMD` directory on the server.

   ----End

3.19.6 Compilation and Installation

**Procedure**

   **Step 1** Use PuTTY to log in to the server as the root user.

   **Step 2** Run the following command to go to the main program installation directory:
   ```
   cd /path/to/ESPRESSOMD
   ```

   **Step 3** Run the following command to decompress the installation package:
   ```
   tar -xvf espresso-4.1.4.tar.gz
   ```
Step 4  Run the following command to go to the directory generated after decompression:

```
cd espresso
```

Step 5  Run the following commands to modify the `cmake/FindFFTW3.cmake` file:

1. Open the `cmake/FindFFTW3.cmake` file.
   ```
   vi cmake/FindFFTW3.cmake
   ```
2. Press i to go to the edit mode. Add the following two lines before line 25:
   ```
   SET(CMAKE_INCLUDE_PATH ${CMAKE_INCLUDE_PATH} "/path/to/FFTW/include")
   SET(CMAKE_LIBRARY_PATH ${CMAKE_LIBRARY_PATH} "/path/to/FFTW/lib")
   ```
3. Press Esc, type `:wq!`, and press Enter to save the file and exit.

Step 6  Create a build directory and build ESPResSo in the directory.

```
mkdir build
```

```
cd build
```

Step 7  Run the following commands to build the software:

```
cmake ..
```

Step 8  Run the following commands to modify the `CMakeCache.txt` file:

1. Open the `CMakeCache.txt` file.
   ```
   vi CMakeCache.txt
   ```
2. Press i to go to the edit mode and add compilation options to line 37 of the `CMakeCache.txt` file:
   ```
   CMAKE_CXX_FLAGS:STRING=-O3 -march=armv8.2-a -mcpu=tsv110 -Wl,-rpath= 
   /path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -L 
   /path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/ 
   lib64/libhpc -lmathlib -lm
   ```
3. Press Esc, type `:wq!`, and press Enter to save the file and exit.

Step 9  Run the following command to configure the environment variables of the `mathlib` library:

```
export LD_LIBRARY_PATH=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/ 
lib64/libhpc:SLD_LIBRARY_PATH
```

Step 10 Run the following commands to perform the compilation and installation:

```
make
```

**NOTE**

After the compilation is successful, the `EspressoCore.so` dynamic library is generated in the `/path/to/ESPRESSOMD/espresso/build/src/core` folder. You can run the `ldd` command to list the dependencies of the dynamic library.

```---End
```

### 3.19.7 Running and Verification

**Procedure**

**Step 1**  Use PuTTY to log in to the server as the root user.
Step 2  Run the following command to run the test case contained in the source package:

```bash
{ time -p mpirun --allow-run-as-root -np 128 -mca pml ucx -mca btl
^vader,tcp,openib,uct -x UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 ./pypresso ../samples/constraints.py > run.log ;} 2>>run.log
```

Step 3  The following is an example output of the log file:

```
(73.85, 103.92, 20.80)
(79.08, 105.94, 25.90)
(67.86, 112.22, 28.00)
(68.77, 113.43, 25.38)
(69.88, 116.07, 21.36)
real 228.98
user 29172.59
sys 55.19
```

----End

3.19.8 More Resources

For more information about ESPResSo, visit the [ESPResSo official website](https://espresso.io).

3.20 MOOSE Framework 1.0.0 Porting Guide (CentOS 7.6)

3.20.1 Introduction

Multiphysics Object Oriented Simulation Environment (MOOSE) is an object-oriented C++ finite element framework for the development of tightly coupled multiphysics solvers. MOOSE makes use of the PETSc non-linear solver package and libmesh to provide the finite element discretization.

MOOSE features high parallelism (greater than 100,000 cores), complete multiphysics, and use of both continuous and discrete Galerkin methods.

For more information about MOOSE, visit the [MOOSE official website](https://mooseproject.org).

Programming language: C/C++

One sentence description: a piece of object-oriented finite element multiphysics environment simulation software.

Open source license: GPL 2.0

Recommended Software Version

MOOSE 1.0.0

3.20.2 Environment Requirements

Hardware Requirements

[Table 3-96](#) lists the hardware requirements.
Table 3-96 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-97 lists the software requirements.

Table 3-97 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOOSE</td>
<td>1.0.0</td>
<td><a href="https://github.com/idaholab/moose/releases/tag/v1.0.0">https://github.com/idaholab/moose/releases/tag/v1.0.0</a></td>
</tr>
<tr>
<td>CMake</td>
<td>3.8.1</td>
<td><a href="https://github.com/Kitware/CMake/archive/v3.8.1.tar.gz">https://github.com/Kitware/CMake/archive/v3.8.1.tar.gz</a></td>
</tr>
<tr>
<td>python</td>
<td>3.8.2</td>
<td><a href="https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tgz">https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tgz</a></td>
</tr>
<tr>
<td>Test cases</td>
<td>hmg</td>
<td>Contained in the software.</td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-98 lists the OS requirements.

Table 3-98 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

3.20.3 Paths for Software Porting

This chapter describes the software installation paths involved in the MOOSE software porting.
### Table 3-99 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/ CMAKE</td>
<td>Installation path of CMake.</td>
<td>The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/ PYTHON</td>
<td>Installation path of Python.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/ MOOSE</td>
<td>Installation path of MOOSE.</td>
<td></td>
</tr>
</tbody>
</table>

### 3.20.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

#### Configure Process

### Table 3-100 Configure process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the open source environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>Setting up the Huawei Kunpeng compiler environment</td>
<td>For details, see section &quot;Installing the Kunpeng GCC&quot; in <a href="#">User Guide (Kunpeng GCC)</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Setting up the Huawei MPI environment</td>
<td>For details, see section &quot;Installation Introduction&quot; in <a href="#">Hyper MPI User Guide</a>.</td>
</tr>
<tr>
<td>4</td>
<td>Installing CMake</td>
<td>See <a href="#">3.20.4.1 Installing CMake</a>.</td>
</tr>
<tr>
<td>5</td>
<td>Installing Python</td>
<td>See <a href="#">3.20.4.2 Installing Python</a>.</td>
</tr>
</tbody>
</table>
3.20.4.1 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following commands to decompress the CMake installation package:
   
   \texttt{cd /path/to/CMAKE}
   
   \texttt{tar -xvf cmake-3.8.1.tar.gz}

Step 3 Run the following command to go to the directory generated after decompression:
   
   \texttt{cd cmake-3.8.1}

Step 4 Run the following command to perform the configuration:
   
   \texttt{./configure --prefix=/path/to/CMAKE}

Step 5 Run the following commands to perform the compilation and installation:
   
   \texttt{make -j16}
   
   \texttt{make install}

Step 6 Run the following command to load the environment variables:
   
   \texttt{export PATH=/path/to/CMAKE/bin:$PATH}

3.20.4.2 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Upload the source package to the \texttt{/path/to/PYTHON} directory.
Step 3 Run the following commands to decompress the Python installation package:
   
   \texttt{cd /path/to/PYTHON}
   
   \texttt{tar -xvf Python-3.8.2.tgz}

Step 4 Run the following command to go to the directory generated after decompression:
   
   \texttt{cd Python-3.8.2}

Step 5 Run the following command to perform the configuration:
   
   \texttt{./configure --prefix=/path/to/PYTHON}

Step 6 Run the following commands to perform the compilation and installation:
   
   \texttt{make -j 16}
   
   \texttt{make install}

Step 7 Run the following command to back up Python:
mv /usr/bin/python /usr/bin/python-bak

**Step 8** Run the following command to create a soft link for Python:

```
ln -s /path/to/PYTHON/bin/python3.8 /usr/bin/python
```

----End

### 3.20.5 Obtaining the Source Code

**Procedure**

1. **Step 1** Run the following command to configure git:
   ```
git config --global http."sslVerify" false
   ```
2. **Step 2** Run the following command to download the MOOSE source code:
   ```
git clone https://github.com/idaholab/moose.git
   ```

----End

### 3.20.6 Compilation and Installation

**Procedure**

1. **Step 1** Use PuTTY to log in to the server as the root user.
2. **Step 2** Run the following command to configure the environment variables:
   ```
   export CC=mpicc
   export CXX=mpicxx
   export F90=mpif90
   export F77=mpif77
   export FC=mpif90
   ```
3. **Step 3** Run the following command to go to the `scripts` directory of MOOSE:
   ```
cd /path/to/MOOSE/moose/scripts
   ```
4. **Step 4** Run the following command to install petsc:
   ```
   .update_and_rebuild_petsc.sh
   ```
5. **Step 5** Run the following command to install libmesh:
   ```
   .update_and_rebuild_libmesh.sh
   ```
6. **Step 6** Run the following command to go to the `test` directory:
   ```
cd /path/to/MOOSE/moose/test
   ```
7. **Step 7** Run the following command to perform the compilation:
   ```
   make -j 16
   ```

----End
3.20.7 Running and Verification

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to go to the test directory:

```bash
cd /path/to/MOOSE/moose/test/tests/preconditioners/hmg
```

**Step 3** Run the following command to perform the test:

```bash
```

**Step 4** Run the following command to restore Python:

```bash
mv /usr/bin/python-bak /usr/bin/python
```

----End

3.21 MPB 1.11.1 and Meep 1.17.1 Porting Guide (CentOS 7.6)

3.21.1 Introduction

MPB is a free and open-source software package for computing the band structures, or dispersion relations, and electromagnetic modes of periodic dielectric structures, on both serial and parallel computers.

Meep is a free finite-difference time-domain (FDTD) simulation software package developed at MIT to model electromagnetic systems, along with the MPB eigenmode package.

For more information about MPB and Meep, visit the [MPB](#) and [Meep](#) official websites.

Programming languages: C++/Python

Brief description: a piece of finite-difference time-domain electromagnetic simulation software.

Open source license: GNU general public license

**Recommended Software Version**

MPB 1.11.1 and Meep 1.17.1
3.21.2 Environment Requirements

Hardware Requirements

Table 3-101 lists the hardware requirements.

Table 3-101 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-102 lists the software requirements.

Table 3-102 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meep</td>
<td>1.17.1</td>
<td><a href="https://codeload.github.com/NanoComp/meep/tar.gz/v1.17.1">https://codeload.github.com/NanoComp/meep/tar.gz/v1.17.1</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.10.1</td>
<td><a href="https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz">https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5-1.10.1.tar.gz</a></td>
</tr>
<tr>
<td>BLAS</td>
<td>3.8.0</td>
<td><a href="http://www.netlib.org/blas/blas.tgz">http://www.netlib.org/blas/blas.tgz</a></td>
</tr>
<tr>
<td>LAPACK</td>
<td>3.9.0</td>
<td><a href="https://codeload.github.com/Reference-LAPACK/lapack/tar.gz/v3.9.0">https://codeload.github.com/Reference-LAPACK/lapack/tar.gz/v3.9.0</a></td>
</tr>
<tr>
<td>HARMINV</td>
<td>1.4.1</td>
<td><a href="https://codeload.github.com/NanoComp/harminv/tar.gz/v1.4.1">https://codeload.github.com/NanoComp/harminv/tar.gz/v1.4.1</a></td>
</tr>
<tr>
<td>FFTW</td>
<td>3.3.8</td>
<td><a href="http://www.fftw.org/fftw-3.3.8.tar.gz">http://www.fftw.org/fftw-3.3.8.tar.gz</a></td>
</tr>
<tr>
<td>libctl</td>
<td>4.5.0</td>
<td><a href="https://codeload.github.com/NanoComp/libctl/tar.gz/v4.5.0">https://codeload.github.com/NanoComp/libctl/tar.gz/v4.5.0</a></td>
</tr>
<tr>
<td>MPB</td>
<td>1.11.1</td>
<td><a href="https://github.com/NanoComp/mpb/archive/v1.11.1.tar.gz">https://github.com/NanoComp/mpb/archive/v1.11.1.tar.gz</a></td>
</tr>
<tr>
<td>libGDSII</td>
<td>0.21</td>
<td><a href="https://codeload.github.com/HomerReid/libGDSII/tar.gz/v0.21">https://codeload.github.com/HomerReid/libGDSII/tar.gz/v0.21</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-103 lists the OS requirements.
### Table 3-103 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 3.21.3 Paths for Software Porting

This chapter describes the software installation paths involved in the Meep software porting.

#### Table 3-104 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/path/to/KPGCC</td>
<td>Installation path of the Kunpeng GCC.</td>
<td>For details, see section &quot;Kunpeng GCC Installation and Operation&quot; in <a href="https://www.centos.org/download/">User Guide (Kunpeng GCC)</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HMPI</td>
<td>Installation path of the Hyper MPI.</td>
<td>For details, see section &quot;Source Code Installation&quot; in <a href="https://www.centos.org/download/">Hyper MPI User Guide</a>.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/SRC</td>
<td>Path of the source package.</td>
<td>The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>4</td>
<td>/path/to/MEEP</td>
<td>Installation path of Meep.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/DEPENDENCIES</td>
<td>Installation path of dependencies.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/CASE</td>
<td>Path for storing test cases.</td>
<td></td>
</tr>
</tbody>
</table>

### 3.21.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.
Configure Process

**Table 3-105 Configure process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the open source environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <strong>HPC Solution Basic Environment Setup Guide</strong>.</td>
</tr>
<tr>
<td>2</td>
<td>Setting up the Huawei Kunpeng compiler environment</td>
<td>For details, see section &quot;Installing the Kunpeng GCC&quot; in <strong>User Guide (Kunpeng GCC)</strong>.</td>
</tr>
<tr>
<td>3</td>
<td>Setting up the Huawei MPI environment</td>
<td>For details, see section &quot;Installation Introduction&quot; in <strong>Hyper MPI User Guide</strong>.</td>
</tr>
<tr>
<td>3</td>
<td>Installing HDF5</td>
<td>See <strong>3.21.4.1 Installing HDF5</strong>.</td>
</tr>
<tr>
<td>4</td>
<td>Installing BLAS</td>
<td>See <strong>3.21.4.2 Installing BLAS</strong>.</td>
</tr>
<tr>
<td>5</td>
<td>Installing LAPACK</td>
<td>See <strong>3.21.4.3 Installing LAPACK</strong>.</td>
</tr>
<tr>
<td>6</td>
<td>Installing HARMINV</td>
<td>See <strong>3.21.4.4 Installing HARMINV</strong>.</td>
</tr>
<tr>
<td>7</td>
<td>Installing FFTW</td>
<td>See <strong>3.21.4.5 Installing FFTW</strong>.</td>
</tr>
<tr>
<td>8</td>
<td>Installing libctl</td>
<td>See <strong>3.21.4.6 Installing libctl</strong>.</td>
</tr>
<tr>
<td>9</td>
<td>Installing MPB</td>
<td>See <strong>3.21.4.7 Installing MPB</strong>.</td>
</tr>
<tr>
<td>10</td>
<td>Installing libGDSII</td>
<td>See <strong>3.21.4.8 Installing libGDSII</strong>.</td>
</tr>
</tbody>
</table>

### 3.21.4.1 Installing HDF5

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the HDF5 installation package:

```bash
cd /path/to/SRC
tar -xvf hdf5-1.10.1.tar
```

**Step 3** Run the following command to go to the directory generated after decompression:

```bash
cd hdf5-1.10.1
```

**Step 4** Run the following command to perform the configuration:

```bash
./configure --enable-parallel --enable-shared --prefix=/path/to/DEPENDENCIES CC=mpicc CXX=mpic++
```
**Step 5** Run the following commands to perform the compilation and installation:

```
make -j
make -j install
```

**Step 6** Run the following commands to configure environment variables:

```
export PATH=/path/to/DEPENDENCIES/bin:$PATH
export LD_LIBRARY_PATH=/path/to/DEPENDENCIES/lib:$LD_LIBRARY_PATH
```

---End

### 3.21.4.2 Installing BLAS

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the BLAS installation package:

```
cd /path/to/SRC
tar -xvf blas.tgz
```

**Step 3** Run the following command to go to the directory generated after decompression:

```
cd BLAS-3.8.0
```

**Step 4** Run the following commands to perform the compilation and installation:

```
`which gfortran` -c -O3 -fomit-frame-pointer -funroll-loops *.f
ar rv libblas.a *.o
```

**Step 5** Run the following command to move the compiled static library to the planned directory:

```
cp libblas.a /path/to/DEPENDENCIES/lib/
```

---End

### 3.21.4.3 Installing LAPACK

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the LAPACK installation package:

```
cd /path/to/SRC
tar -xvf lapack-3.9.0.tar.gz
```

**Step 3** Run the following command to go to the directory generated after decompression:

```
cd lapack-3.9.0
```

**Step 4** Run the following commands to perform the compilation and installation:
Step 5 Run the following command to move the compiled static library to the planned directory:

```bash
cp liblapack.a /path/to/DEPENDENCIES/lib/
```

---End

### 3.21.4.4 Installing HARMINV

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the HARMINV installation package:

```bash
cd /path/to/SRC
tar -xvf harminv-1.4.1.tar.gz
```

**Step 3** Run the following command to go to the directory generated after decompression:

```bash
cd harminv-1.4.1
```

**Step 4** Run the following command to perform the configuration:

```bash
sh autogen.sh --prefix=/path/to/DEPENDENCIES --enable-shared LDFLAGS="-L/path/to/DEPENDENCIES/lib"
```

**Step 5** Run the following commands to perform the compilation and installation:

```bash
make -j
make -j install
```

---End

### 3.21.4.5 Installing FFTW

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the FFTW installation package:

```bash
cd /path/to/SRC
tar -xvf fftw-3.3.8.tar.gz
```

**Step 3** Run the following command to go to the directory generated after decompression:

```bash
cd fftw-3.3.8
```

**Step 4** Run the following command to execute the `bootstrap.sh` script:

```bash
./bootstrap.sh
```
Step 5 Run the following command to perform the configuration:

```
./configure --prefix=/path/to/DEPENDENCIES --enable-shared --enable-static --enable-fma --enable-neon --enable-mpi
```

Step 6 Run the following commands to perform the compilation and installation:

```
make -j
make -j install
```

3.21.4.6 Installing libctl

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the libctl installation package:

```
cd /path/to/SRC
tar -xvf libctl-4.5.0.tar.gz
```

**Step 3** Run the following command to go to the directory generated after decompression:

```
cd libctl-4.5.0
```

**Step 4** Run the following command to install the system dependencies:

```
yum install -y guile-devel
```

**Step 5** Run the following command to perform the configuration:

```
sh autogen.sh --enable-shared --prefix=/path/to/DEPENDENCIES
```

**Step 6** Run the following commands to perform the compilation and installation:

```
make -j
make -j install
```

3.21.4.7 Installing MPB

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the MPB installation package:

```
cd /path/to/SRC
tar -xvf mpb-1.11.1.tar.gz
```

**Step 3** Run the following command to go to the directory generated after decompression:

```
cd mpb-1.11.1
```
Step 4  Run the following command to install the system packages:

```
yum install -y guile-devel
```

Step 5  Run the following command to configure the environment variables of the `mathlib` library:

```
export LD_LIBRARY_PATH=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc:LD_LIBRARY_PATH
```

Step 6  Run the following command to perform the serial configuration:

```
sh autogen.sh --prefix=/path/to/DEPENDENCIES --enable-shared CC=mpicc
LDFLAGS="-L/path/to/DEPENDENCIES\lib -Wl,-rpath=/path/to/KPGCC\gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -L/path/to/KPGCC\gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -lmathlib -lm" CPPFLAGS="-I/path/to/DEPENDENCIES\include" CFLAGS="-O3 -march=armv8.2-a -mtune=tsv110" --with-hermitian-eps --with-libctl=/path/to/DEPENDENCIES\share/libctl/
```

Step 7  Run the following commands to perform the serial compilation and installation:

```
make -j
make -j install
```

Step 8  (Optional) Run the following command to perform the parallel configuration:

```
sh autogen.sh --prefix=/path/to/DEPENDENCIES --enable-shared CC=mpicc
LDFLAGS="-L/path/to/DEPENDENCIES\lib -Wl,-rpath=/path/to/KPGCC\gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -L/path/to/KPGCC\gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -lmathlib -lm" CPPFLAGS="-I/path/to/DEPENDENCIES\include" CFLAGS="-O3 -march=armv8.2-a -mtune=tsv110" --with-hermitian-eps --with-mpi --with-libctl=/path/to/DEPENDENCIES\share/libctl/
```

Step 9  (Optional) Run the following commands to perform the parallel compilation and installation:

```
make -j
make -j install
```

**NOTE**

Meep can only be linked to the serial version of MPB. For the linkage to the parallel version of MPB, you need to perform the compilation and installation.

```
--------End
```

### 3.21.4.8 Installing libGDSII

**Procedure**

**Step 1**  Use PuTTY to log in to the server as the root user.

**Step 2**  Run the following commands to decompress the libGDSII installation package:

```
cd /path/to/SRC
```
Step 3 Go to the directory generated after decompression:

```
cd libGDSII-0.21
```

Step 4 Run the following command to perform the configuration:

```
sh autogen.sh --prefix=/path/to/DEPENDENCIES
```

Step 5 Run the following commands to perform the compilation and installation:

```
make -j
make -j install
```

---End

3.21.5 Obtaining the Source Code

Procedure

Step 1 Download the Meep installation package `meep-1.17.1.tar.gz`.

Download address: [https://codeload.github.com/NanoComp/meep/tar.gz/v1.17.1](https://codeload.github.com/NanoComp/meep/tar.gz/v1.17.1)

Step 2 Use an SFTP tool to upload the Meep installation package to the `/path/to/SRC` directory on the server.

---End

3.21.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the Meep installation package:

```
cd /path/to/SRC
tar -xvf meep-1.17.1.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd meep-1.17.1
```

Step 4 Run the following command to perform the configuration:

```
sh autogen.sh --prefix=/path/to/MEEP --enable-shared --with-mpi --with-openmp PYTHON=python3 MPICC=`which mpicc` MPICXX=`which mpic++` LDFLAGS="-L/path/to/DEPENDENCIES|lib -Wl,-rpath=/path/to/KPGCC` gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc/ -L/path/to/KPGCC` gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc -lm -lmathlib -lm` CPPFLAGS="-I/path/to/DEPENDENCIES|include" CFLAGS="-O3 -march=armv8.2-a -mtune=tsv110" --with-libctl=/path/to/DEPENDENCIES|share/libctl/
```

Step 5 Run the following commands to perform the compilation and installation:
make -j
make -j install

**Step 6** Run the following commands to view the generated executable file:

```bash
ll /path/to/MEEP/bin
```

-`-rwxr-xr-x 1 root root 1139520 Feb 3 11:43 meep

----End

### 3.21.7 Running and Verification

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to configure the environment variables of the Meep executable file:

```bash
export PATH=/path/to/MEEP/bin:$PATH
```

**Step 3** Run the following command to copy the test case file to the planned directory:

```bash
cp /path/to/SRC/meep-1.17.1/scheme/examples/mie-scattering.ctl /path/to/CASE
```

**Step 4** Run the following command to go to the test case directory:

```bash
cd /path/to/CASE
```

**Step 5** Run the following command to execute Meep:

```bash
mpirun --allow-run-as-root -mca pml ucx -mca btl ^vader,tcp,openib,uct -x UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 -np 128 meep mie-scattering.ctl | tee mie.out
```

**Step 6** Run the following commands to extract the input data and far-field data to a separate file:

```bash
grep flux1: mie.out |cut -d, -f2 -> input.dat
grep flux2: mie.out |cut -d, -f2 -> scatt.dat
```

**Step 7** After the commands are executed, check the value of *Elapsed run time* in the last line of the *mie.out* file. A smaller value indicates a higher performance.

----End

### 3.21.8 More Resources

MPB official website:


Meep official website:

3.22 Octopus 10.3 Porting Guide (CentOS 7.6)

3.22.1 Introduction

Octopus is a pseudopotential real space package that simulates the electron-ion dynamics of one, two, and three-dimensional finite systems subject to time-dependent electromagnetic fields. The program is based on time-dependent density functional theory (TDDFT) in the Kohn-Sham scheme. Octopus can calculate static and dynamic polarizabilities and first hyperpolarizabilities, static magnetic susceptibilities, absorption spectra, and perform molecular dynamics simulations with Ehrenfest and Car–Parrinello methods.

For more information about Octopus, visit the Octopus official website.

Programming languages: C/C++

Brief description: a piece of quantum mechanics simulation software.

Recommended Software Version

Octopus 10.3

3.22.2 Environment Requirements

Hardware Requirements

Table 3-106 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-107 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFTW</td>
<td>3.3.8</td>
<td><a href="http://www.fftw.org/fftw-3.3.8.tar.gz">http://www.fftw.org/fftw-3.3.8.tar.gz</a></td>
</tr>
<tr>
<td>GSL</td>
<td>2.6</td>
<td><a href="http://mirrors.ustc.edu.cn/gnu/gsl/gsl-2.6.tar.gz">http://mirrors.ustc.edu.cn/gnu/gsl/gsl-2.6.tar.gz</a></td>
</tr>
<tr>
<td>Libxc</td>
<td>4.3.4</td>
<td><a href="http://forge.abinit.org/fallbacks/libxc-4.3.4.tar.gz">http://forge.abinit.org/fallbacks/libxc-4.3.4.tar.gz</a></td>
</tr>
</tbody>
</table>
### OS Requirements

*Table 3-108* lists the OS requirements.

**Table 3-108 OS requirements**

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 3.22.3 Paths for Software Porting

This chapter describes the software installation paths involved in the Octopus software porting.

**Table 3-109 Paths for software porting**

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>/path/to/KPGCC</code></td>
<td>Installation path of the Kunpeng GCC.</td>
<td>For details, see section &quot;Kunpeng GCC Installation and Operation&quot; in <a href="#">User Guide (Kunpeng GCC)</a>.</td>
</tr>
<tr>
<td>2</td>
<td><code>/path/to/HMPI</code></td>
<td>Installation path of the Hyper MPI.</td>
<td>For details, see section &quot;Source Code Installation&quot; in <a href="#">Hyper MPI User Guide</a>.</td>
</tr>
<tr>
<td>3</td>
<td><code>/path/to/FFTW</code></td>
<td>Installation path of FFTW.</td>
<td>The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>4</td>
<td><code>/path/to/GSL</code></td>
<td>Installation path of GSL.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><code>/path/to/LIBXC</code></td>
<td>Installation path of Libxc.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><code>/path/to/OCTOPUS</code></td>
<td>Installation path of Octopus.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td><code>/path/to/TESTCASE</code></td>
<td>Path for storing test cases.</td>
<td></td>
</tr>
</tbody>
</table>
3.22.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

Table 3-110 Configure process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the open source environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Setting up the Huawei Kunpeng compiler environment</td>
<td>For details, see section &quot;Installing the Kunpeng GCC&quot; in User Guide (Kunpeng GCC).</td>
</tr>
<tr>
<td>3</td>
<td>Setting up the Huawei MPI environment</td>
<td>For details, see section &quot;Installation Introduction&quot; in Hyper MPI User Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing LAPACK</td>
<td>See 3.22.4.1 Installing Dependencies.</td>
</tr>
<tr>
<td>3</td>
<td>Installing BLAS</td>
<td>See 3.22.4.1 Installing Dependencies.</td>
</tr>
<tr>
<td>4</td>
<td>Installing FFTW</td>
<td>See 3.22.4.2 Installing FFTW.</td>
</tr>
<tr>
<td>5</td>
<td>Installing GSL</td>
<td>See 3.22.4.3 Installing GSL.</td>
</tr>
<tr>
<td>6</td>
<td>Installing Libxc</td>
<td>See 3.22.4.4 Installing Libxc.</td>
</tr>
</tbody>
</table>

3.22.4.1 Installing Dependencies

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies:

```bash
yum install lapack-devel blas-devel -y
```

----End

3.22.4.2 Installing FFTW

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the FFTW installation package:

```bash
tar -xvf fftw-3.3.8.tar.gz
```
Step 3 Run the following command to go to the directory generated after decompression:

cd fftw-3.3.8

Step 4 Run the following command to perform the configuration:

./configure --prefix=/path/to/FFTW --enable-shared --enable-static --enable-fma --enable-neon

Step 5 Run the following commands to perform the compilation and installation:

make
make install

Step 6 Run the following commands to configure environment variables:

export LIBRARY_PATH=/path/to/FFTW/lib:$LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/FFTW/lib:$LD_LIBRARY_PATH

3.22.4.3 Installing GSL

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the GSL installation package:

tar -xvf gsl-2.6.tar.gz

Step 3 Run the following command to go to the directory generated after decompression:

cd gsl-2.6

Step 4 Run the following command to perform the configuration:

./configure --prefix=/path/to/GSL/gsl

Step 5 Run the following commands to perform the compilation:

make -j 32
make install

Step 6 Run the following commands to configure environment variables:

export LIBRARY_PATH=/path/to/GSL/gsl/lib:$LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/GSL/gsl/lib:$LD_LIBRARY_PATH

----End

3.22.4.4 Installing Libxc

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2  Run the following command to decompress the Libxc installation package:

```
tar -xvf libxc-4.3.4.tar.gz
```

Step 3  Run the following command to go to the directory generated after decompression:

```
cd libxc-4.3.4
```

Step 4  Run the following command to perform the configuration:

```
./configure FC=mpifort CC=mpicc --prefix=/path/to/LIBXC/libxc434
```

Step 5  Run the following commands to perform the compilation:

```
make
make install
```

3.22.5 Obtaining the Source Code

Procedure

Step 1  Download the Octopus installation package `octopus-10.3.tar.gz`.


Step 2  Use an SFTP tool to upload the Octopus installation package to the `/path/to/OCTOPUS` directory on the server.

3.22.6 Compilation and Installation

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to decompress the Octopus package:

```
tar -xvf octopus-10.3.tar.gz
```

Step 3  Run the following command to go to the directory generated after decompression:

```
cd octopus-10.3
```

Step 4  Run the following command to perform the configuration:

```
```

Step 5  Run the following commands to perform the compilation and installation:

```
make -j 32
```
Step 6 Run the following command to configure the environment variables:

```
export PATH=/path/to/OCTOPUS/bin:$PATH
```

----End

3.22.7 Running and Verification

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to go to the test case directory:

```
cd /path/to/TESTCASE
```

**Step 3** Run the following commands to create an input file named `inp` for calculating the ground state of the hydrogen atom, and then edit the file:

1. Open the `inp` file.

   ```
   vi inp
   ```

2. Press `i` to go to the edit mode and add the following content:

   ```
   CalculationMode = gs
   %Coordinates
   'H'|0|0|0
   %
   ```

3. Press `Esc`, type `:wq!`, and press `Enter` to save the file and exit.

**Step 4** Run the following command to execute the test case file:

```
mpirun --allow-run-as-root -mca pml ucx -mca btl ^vader,tcp,openib,uct -x
UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -np 128 -x
UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 octopus
```

**Step 5** Run the following command to check the calculation result:

```
cat static/info
```
3.22.8 More Resources

Octopus official website:


3.23 Psi4 1.3.2 Porting Guide (CentOS 7.6)

3.23.1 Introduction

Psi4 is an open source suite of ab initio quantum chemistry programs designed for efficient, high-accuracy simulations of molecular properties. It routinely performs computations with more than 2500 basis functions on multi-core machines.

With computationally demanding portions written in C++, exports of many C++ classes into Python via Pybind11, and a flexible Python driver, Psi4 strives to be friendly to both users and developers.

For more information about Psi4, visit the Psi4 official website.

Programming languages: C++/Python

Brief description: an open-source quantum chemistry software package.

Open source license: GNU general public license

Recommended Software Version

Psi4 1.3.2
3.23.2 Environment Requirements

Hardware Requirements

Table 3-111 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-112 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psi4</td>
<td>1.3.2</td>
<td><a href="https://github.com/psi4/psi4/archive/v1.3.2.tar.gz">https://github.com/psi4/psi4/archive/v1.3.2.tar.gz</a></td>
</tr>
<tr>
<td>CMake</td>
<td>3.8.1</td>
<td><a href="https://github.com/Kitware/CMake/archive/v3.8.1.tar.gz">https://github.com/Kitware/CMake/archive/v3.8.1.tar.gz</a></td>
</tr>
<tr>
<td>OpenBLAS</td>
<td>0.3.6</td>
<td><a href="https://github.com/xianyi/OpenBLAS/archive/v0.3.6.tar.gz">https://github.com/xianyi/OpenBLAS/archive/v0.3.6.tar.gz</a></td>
</tr>
<tr>
<td>gau2grid</td>
<td>1.3.0</td>
<td><a href="https://github.com/dgasmith/gau2grid/archive/v1.3.0.tar.gz">https://github.com/dgasmith/gau2grid/archive/v1.3.0.tar.gz</a></td>
</tr>
<tr>
<td>libint</td>
<td>release-1-2-0</td>
<td><a href="https://github.com/evaleev/libint/archive/release-1-2-0.tar.gz">https://github.com/evaleev/libint/archive/release-1-2-0.tar.gz</a></td>
</tr>
<tr>
<td>libxc</td>
<td>4.3.0</td>
<td><a href="https://gitlab.com/libxc/libxc/-/archive/4.3.0/libxc-4.3.0.tar.gz">https://gitlab.com/libxc/libxc/-/archive/4.3.0/libxc-4.3.0.tar.gz</a></td>
</tr>
<tr>
<td>pybind11</td>
<td>2.6.2</td>
<td><a href="https://github.com/pybind/pybind11/archive/v2.6.2.tar.gz">https://github.com/pybind/pybind11/archive/v2.6.2.tar.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-113 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
</tbody>
</table>
### 3.23.3 Paths for Software Porting

This chapter describes the software installation paths involved in the Psi4 software porting.

**Table 3-114 Paths for software porting**

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/path/to/KPGCC</td>
<td>Installation path of the Kunpeng GCC.</td>
<td>For details, see section &quot;Kunpeng GCC Installation and Operation&quot; in User Guide (Kunpeng GCC).</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HMPI</td>
<td>Installation path of the Hyper MPI.</td>
<td>For details, see section &quot;Source Code Installation&quot; in Hyper MPI User Guide.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/PSI4</td>
<td>Installation path of Psi4.</td>
<td>The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>4</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/GAU2GRID</td>
<td>Installation path of gau2grid.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>/path/to/LIBINT</td>
<td>Installation path of Libint.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>/path/to/LIBXC</td>
<td>Installation path of Libxc.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>/path/to/PYBIND11</td>
<td>Installation path of pybind11.</td>
<td></td>
</tr>
</tbody>
</table>

### 3.23.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages have been uploaded to the server using an SFTP tool.
Configure Process

Table 3-115 Configure process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the open source environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Setting up the Huawei Kunpeng compiler environment</td>
<td>For details, see section &quot;Installing the Kunpeng GCC&quot; in <em>User Guide (Kunpeng GCC)</em>.</td>
</tr>
<tr>
<td>3</td>
<td>Setting up the Huawei MPI environment</td>
<td>For details, see section &quot;Installation Introduction&quot; in <em>Hyper MPI User Guide</em>.</td>
</tr>
<tr>
<td>3</td>
<td>Installing system dependencies</td>
<td>See 3.23.4.1 Installing System Dependencies.</td>
</tr>
<tr>
<td>4</td>
<td>Installing CMake</td>
<td>See 3.23.4.2 Installing CMake.</td>
</tr>
<tr>
<td>5</td>
<td>Installing OpenBLAS</td>
<td>See 3.23.4.3 Installing OpenBLAS.</td>
</tr>
<tr>
<td>6</td>
<td>Installing gau2grid</td>
<td>See 3.23.4.4 Installing gau2grid.</td>
</tr>
<tr>
<td>7</td>
<td>Installing Libint</td>
<td>See 3.23.4.5 Installing Libint.</td>
</tr>
<tr>
<td>8</td>
<td>Installing Libxc</td>
<td>See 3.23.4.6 Installing Libxc.</td>
</tr>
<tr>
<td>9</td>
<td>Installing pybind11</td>
<td>See 3.23.4.7 Installing pybind11.</td>
</tr>
</tbody>
</table>

3.23.4.1 Installing System Dependencies

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to install the system dependencies:

```bash
yum install -y numpy python3 python3-devel python3-libs python3-pip
pip3 install QCElemental QCEngine deepdiff
```

---End

3.23.4.2 Installing CMake

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.
Step 2  Run the following commands to decompress the CMake installation package:

    cd /path/to/CMAKE
    tar -xvf cmake-3.8.1.tar.gz

Step 3  Run the following command to go to the directory generated after decompression:

    cd cmake-3.8.1

Step 4  Run the following command to perform the configuration:

    mkdir install
    ./configure --prefix=/path/to/CMAKE/cmake-3.8.1/install

Step 5  Run the following command to perform the compilation and installation:

    make -j
    make install

Step 6  Run the following command to configure environment variables:

    export PATH=/path/to/CMAKE/cmake-3.8.1/install/bin:$PATH

---End

3.23.4.3 Installing OpenBLAS

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to decompress the OpenBLAS installation package:

    cd /path/to/OPENBLAS
    tar -xvf OpenBLAS-0.3.6.tar.gz

Step 3  Run the following command to go to the directory generated after decompression:

    cd OpenBLAS-0.3.6

Step 4  Run the following commands to configure environment variables:

    export CC=`which gcc`
    export CXX=`which g++`
    export FC=`which gfortran`

Step 5  Run the following commands to modify the Makefile.system file:

1. Open the Makefile.system file.
   vi Makefile.system
2. Type :1161 to go to line 1161 and press i to go to the edit mode.

Before the modification:

    COMMON_OPT = -O2

After the modification:
3. Press Esc, type :wq!, and press Enter to save the file and exit.

**Step 6** Run the following commands to perform the compilation and installation:

- `mkdir install`
- `make -j`
- `make PREFIX=/path/to/OPENBLAS/OpenBLAS-0.3.6/install install`

**Step 7** Run the following commands to configure environment variables:

- `export MATH_ROOT=/path/to/OPENBLAS/OpenBLAS-0.3.6/install`

---End

### 3.23.4.4 Installing gau2grid

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the gau2grid installation package:

- `cd /path/to/GAU2GRID`
- `tar -xvf gau2grid-1.3.0.tar.gz`

**Step 3** Run the following command to go to the directory generated after decompression:

- `cd gau2grid-1.3.0`

**Step 4** Run the following commands to perform the compilation:

- `mkdir /path/to/GAU2GRID/build`
- `cmake -H. -B /path/to/GAU2GRID/build -DCMAKE_INSTALL_PREFIX=/path/to/GAU2GRID/install`

**Step 5** Run the following commands to perform the installation:

- `cd /path/to/GAU2GRID/build`
- `make -j`
- `make install`

---End

### 3.23.4.5 Installing Libint

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the Libint installation package:

- `cd /path/to/LIBINT`
- `tar -xvf libint-release-1-2-0.tar.gz`
Step 3 Run the following command to go to the directory generated after decompression:
   cd libint-release-1-2-0

Step 4 Run the following commands to perform the compilation:
   mkdir /path/to/LIBINT/build
   cmake -H. -B/path/to/LIBINT/build -DCMAKE_INSTALL_PREFIX=/path/to/LIBINT/install

Step 5 Run the following commands to perform the installation:
   cd /path/to/LIBINT/build
   make -j
   make install
   ----End

3.23.4.6 Installing Libxc

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the Libxc installation package:
   cd /path/to/LIBXC
   tar -xvf libxc-4.3.0.tar.gz

Step 3 Run the following command to go to the directory generated after decompression:
   cd libxc-4.3.0

Step 4 Run the following commands to perform the compilation:
   mkdir /path/to/LIBXC/build
   cmake -H. -B/path/to/LIBXC/build -DCMAKE_INSTALL_PREFIX=/path/to/LIBXC/install

Step 5 Run the following commands to perform the installation:
   cd /path/to/LIBXC/build
   make -j
   make install
   ----End

3.23.4.7 Installing pybind11

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2  Run the following commands to decompress the pybind11 installation package:
   `cd /path/to/PYBIND11`
   `tar -xvf pybind11-2.6.2.tar.gz`

Step 3  Run the following command to go to the directory generated after decompression:
   `cd pybind11-2.6.2`

Step 4  Run the following commands to perform the compilation:
   `mkdir /path/to/PYBIND11/build`
   `cmake -H. -B /path/to/PYBIND11/build -DCMAKE_INSTALL_PREFIX=/path/to/PYBIND11/install`

Step 5  Run the following commands to perform the installation:
   `cd /path/to/PYBIND11/build`
   `make -j`
   `make install`

----End

3.23.5 Obtaining the Source Code

Procedure

Step 1  Download the Psi4 installation package `psi4-1.3.2.tar.gz`.
   Download address: [https://github.com/psi4/psi4/archive/v1.3.2.tar.gz](https://github.com/psi4/psi4/archive/v1.3.2.tar.gz)

Step 2  Use an SFTP tool to upload the Psi4 installation package to the `/path/to/PSI4` directory on the server.

----End

3.23.6 Compilation and Installation

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to decompress the Psi4 installation package:
   `cd /path/to/PSI4`
   `tar -xvf psi4-1.3.2.tar.gz`

Step 3  Run the following command to go to the directory generated after decompression:
   `cd psi4-1.3.2`

Step 4  Run the following commands to perform the compilation:
   `mkdir build install`
Step 5 Run the following commands to perform the installation:

```bash
cd /path/to/PSI4/build
make -j
make install
```

Step 6 Run the following commands to view the generated executable file:

```bash
ll /path/to/PSI4/install/bin
```

```
-rw-r-xr-x 1 root root 12139 Mar  9 10:47 psi4
```

----End

### 3.23.7 Running and Verification

#### Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to configure the environment variables of the Psi4 executable file:

```bash
export PATH=/path/to/PSI4/install/bin:$PATH
```

**Step 3** Run the following command to go to the test case directory:

```bash
cd /path/to/PSI4/install/share/psi4/samples/opt-multi-frozen-dimer-c2h
```

**Step 4** Run the following command to run Psi4:

```bash
psi4 input.dat
```

**Step 5** After the running is complete, check the value of **Wall Time** in the **timer.dat** file. A smaller value indicates a higher performance.

----End

### 3.23.8 More Resources

Psi4 official website:

`http://ab-initio.mit.edu/wiki/index.php/PSI4`

### 3.24 SIESTA 4.0.2 Porting Guide (CentOS 7.6)
3.24.1 Introduction

SIESTA is both a method and its computer program implementation, to perform efficient electronic structure calculations and ab initio molecular dynamics simulations of molecules and solids.

For more information about SIESTA, visit the SIESTA official website.

Programming languages: C/C++

Brief description: a piece of material calculation simulation software.

Recommended Software Version

siesta-4.0.2

3.24.2 Environment Requirements

Hardware Requirements

Table 3-116 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-117 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIESTA</td>
<td>4.0.2</td>
<td><a href="https://launchpadlibrarian.net/379220821/siesta-4.0.2.tar.gz">https://launchpadlibrarian.net/379220821/siesta-4.0.2.tar.gz</a></td>
</tr>
<tr>
<td>ScaLAPACK CK</td>
<td>2.0.2</td>
<td><a href="http://www.netlib.org/scalapack/scalapack-2.0.2.tgz">http://www.netlib.org/scalapack/scalapack-2.0.2.tgz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-118 lists the OS requirements.
Table 3-118 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

3.24.3 Paths for Software Porting

This chapter describes the software installation paths involved in the SIESTA software porting.

Table 3-119 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/path/to/KPGCC</td>
<td>Installation path of the Kunpeng GCC.</td>
<td>For details, see section &quot;Kunpeng GCC Installation and Operation&quot; in <a href="https://www.centos.org/download/">User Guide (Kunpeng GCC)</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HMPI</td>
<td>Installation path of the Hyper MPI.</td>
<td>For details, see section &quot;Source Code Installation&quot; in <a href="https://www.centos.org/download/">Hyper MPI User Guide</a>.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/SIESTA</td>
<td>Installation path of SIESTA.</td>
<td>The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>4</td>
<td>/path/to/SCALAPACK</td>
<td>Installation path of ScaLAPACK.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/TESTCASE</td>
<td>Path for storing test cases.</td>
<td></td>
</tr>
</tbody>
</table>

3.24.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.
Configure Process

Table 3-120 Configure process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the open source environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Setting up the Huawei Kunpeng compiler environment</td>
<td>For details, see section &quot;Installing the Kunpeng GCC&quot; in User Guide (Kunpeng GCC).</td>
</tr>
<tr>
<td>3</td>
<td>Setting up the Huawei MPI environment</td>
<td>For details, see section &quot;Installation Introduction&quot; in Hyper MPI User Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing LAPACK</td>
<td>See 3.24.4.1 Installing Dependencies.</td>
</tr>
<tr>
<td>3</td>
<td>Installing BLAS</td>
<td>See 3.24.4.1 Installing Dependencies.</td>
</tr>
<tr>
<td>4</td>
<td>Installing ScaLAPACK</td>
<td>See 3.24.4.2 Installing ScaLAPACK.</td>
</tr>
</tbody>
</table>

3.24.4.1 Installing Dependencies

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to install dependencies:

```bash
yum install lapack-devel blas-devel -y
```

----End

3.24.4.2 Installing ScaLAPACK

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the ScaLAPACK installation package:

```bash
tar -xvf scalapack-2.0.2.tgz
```

**Step 3** Run the following command to go to the directory generated after decompression:

```bash
cd scalapack-2.0.2
```

**Step 4** Run the following command to generate an SLmake.inc file:

```bash
cp SLmake.inc.example SLmake.inc
```

**Step 5** Run the following command to perform the compilation and installation:

```bash
make
```
Step 6 Run the following commands to copy the generated link library to the installation directory:

```
mkdir -p /path/to/SCALAPACK
cp libscalapack.a /path/to/SCALAPACK
```

---End

3.24.5 Obtaining the Source Code

Procedure

Step 1 Download the SIESTA installation package `siesta-4.0.2.tar.gz`.

Download address: [https://launchpadlibrarian.net/379220821/siesta-4.0.2.tar.gz](https://launchpadlibrarian.net/379220821/siesta-4.0.2.tar.gz)

Step 2 Use an SFTP tool to upload the SIESTA installation package to the `/path/to/SIESTA` directory on the server.

---End

3.24.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the SIESTA package:

```
tar -xvf siesta-4.0.2.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd siesta-4.0.2
```

Step 4 Run the following commands to create a directory:

```
mkdir kpgcc_hmpi_compiler
```

Step 5 Run the following commands to go to the directory and run the script:

```
cd kpgcc_hmpi_compiler
bash ../Src/obj_setup.sh
```

Step 6 Run the following command to perform the configuration:

```
..Src/configure CC=mpicc CFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' FC=mpifort FCFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' --enable-mpi --build=arm-linux --prefix=/path/to/SIESTA/siesta-4.0.2/kpgcc_hmpi_compiler
```

Step 7 Run the following commands to modify the `arch.make` file:

1. Open the `arch.make` file.
   
   `vi arch.make`

2. Press `i` to go to the edit mode and modify lines 38, 39, and 45 of the `arch.make` file.
3. Press Esc, type :wq!, and press Enter to save the file and exit.

**Step 8** Run the following command to perform the compilation and installation:

```
make
```

**Step 9** Run the following command to configure the environment variables:

```
export PATH=/path/to/SIESTA/:SPATH
```

----End

### 3.24.7 Running and Verification

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to copy the Tests directory of the source package to the test case directory:

```
cd /path/to/SIESTA
cp -r ./siesta-4.0.2/Tests /path/to/TESTCASE
```

**Step 3** Run the following command to go to the test case directory:

```
cd /path/to/TESTCASE/Tests/born
```

**Step 4** Run the following command to execute the test case file:

```
make SIESTA="mpirun --allow-run-as-root -np 128 siesta"
```

If the born.out and work files are generated, the execution is successful.

----End

### 3.24.8 Troubleshooting

**Problem 1: An error is reported when performing step 7 of SIESTA compilation and installation**

**Symptom:**

During the SIESTA compilation and installation, an error is reported in step 7:

```
configure: error: cannot guess build type; you must specify one
make: *** [FoX/config] Error 1
```

**Possible Causes:**

checking build system type... /Oceanfile/RR3/kunpeng/application/siesta/siesta-4.0.2/Src/FoX/config/config guess: unable to guess system type
This script, last modified 2006-12-08, has failed to recognize the operating system you are using. It is advised that you download the most up to date version of the config scripts from http://savannah.gnu.org/cgi-bin/viewcvs/*checkout*/config/config/config.guess and http://savannah.gnu.org/cgi-bin/viewcvs/*checkout*/config/config/config.sub

Procedure:

1. Perform the following operations:
   - http://savannah.gnu.org/cgi-bin/viewcvs/*checkout*/config/config/config.guess
   - http://savannah.gnu.org/cgi-bin/viewcvs/*checkout*/config/config/config.sub

2. Replace the config.guess and config.sub files in the /path/to/SIESTA/siesta-4.0.2/SRC/Fox/config directory. Run the make clean command and run the make command for compilation again.

3.24.9 More Resources

SIESTA official website:

https://departments.icmab.es/leem/siesta/

3.25 Wannier90 3.1.0 Porting Guide (CentOS 7.6)

3.25.1 Introduction

Wannier90 is a computer package written in Fortran90. It is used to obtain maximally-localised Wannier functions in order to calculate bandstructures, Fermi surfaces, dielectric properties, sparse Hamiltonians and many things besides.

Many electronic structure codes have an interface to Wannier90, including Quantum ESPRESSO, Abinit, VASP, Siesta, Wien2k, Fleur, OpenMX, and GPAW. There are several post-processing codes that are able to use the output of Wannier90 for further analysis and calculation.

Programming language: Fortran

Brief description: Wannier90 is a computer package, written in Fortran90, used to obtain maximally-localised Wannier functions.

Open source license: GPL-2.0

Recommended Software Version

wannier90-3.1.0

3.25.2 Environment Requirements

Hardware Requirements

Table 3-121 lists the hardware requirements.
Table 3-121 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-122 lists the software requirements.

Table 3-122 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wannier90</td>
<td>3.1.0</td>
<td><a href="https://codeload.github.com/wannier-developers/wannier90/tar.gz/v3.1.0">https://codeload.github.com/wannier-developers/wannier90/tar.gz/v3.1.0</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-123 lists the OS requirements.

Table 3-123 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

3.25.3 Paths for Software Porting

This chapter describes the software installation paths involved in the Wannier90 software porting.

Table 3-124 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/path/to/KPGCC</td>
<td>Installation path of the Kunpeng GCC.</td>
<td>For details, see section “Kunpeng GCC Installation and Operation” in User Guide (Kunpeng GCC).</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HMPI</td>
<td>Installation path of the Hyper MPI.</td>
<td>For details, see section “Source Code Installation” in Hyper MPI User Guide.</td>
</tr>
</tbody>
</table>
### 3.25.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages have been uploaded to the server using an SFTP tool.

**Configure Process**

**Table 3-125 Configure process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the open source environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Setting up the Huawei Kunpeng compiler environment</td>
<td>For details, see section &quot;Installing the Kunpeng GCC&quot; in User Guide (Kunpeng GCC).</td>
</tr>
<tr>
<td>3</td>
<td>Setting up the Huawei MPI environment</td>
<td>For details, see section &quot;Installation Introduction&quot; in Hyper MPI User Guide.</td>
</tr>
</tbody>
</table>

### 3.25.4.1 Installing the Dependency Libraries and Tools Using Yum

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to install the LAPACK and BLAS libraries as well as the gnuplot interactive graphing tool.

```
yum install lapack* blas* gnuplot -y
```
The LAPACK and BLAS libraries are installed in /usr/lib64 by default. The gnuplot is installed in /usr by default.

**Step 3** Run the following commands to configure gnuplot environment variables:

```bash
export GNUTERM=dumb
export GNUPLOT=/usr
export PATH=/usr/bin:$PATH
export MANPATH=/usr/share/man/man1:$MANPATH
```

### 3.25.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the Wannier90 installation package `wannier90-3.1.0.tar.gz`.

Download address: [https://codeload.github.com/wannier-developers/wannier90/tar.gz/v3.1.0](https://codeload.github.com/wannier-developers/wannier90/tar.gz/v3.1.0)

**Step 2** Use an SFTP tool to upload the Wannier90 installation package to the `/path/to/WANNIER90` directory on the server.

### 3.25.6 Compilation and Installation

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to go to the main program installation directory:

```bash
cd /path/to/WANNIER90
```

**Step 3** Run the following command to decompress the main program installation package:

```bash
tar -xvf wannier90-3.1.0.tar.gz
```

**Step 4** Run the following command to go to the directory generated after decompression:

```bash
cd wannier90-3.1.0
```

**Step 5** Run the following command to perform build Wannier90:

```bash
cp ./config/make.inc.gfort ./make.inc
```

**Step 6** Run the following commands to modify the `make.inc` file:

1. Open the `make.inc` file.

   ```bash
   vi make.inc
   ```
2. Press i to go to the edit mode.
   - Uncomment lines 7 and 8, and modify line 8 as follows:
     \[ \text{MPIF90 = mpif90} \]
   - Modify line 10 as follows:
     \[ \text{FCOPTS = -O3 -march=armv8.2-a -mcpu=tsv110} \]
   - Uncomment line 21 and modify it as follows:
     \[ \text{LIBDIR = /usr/lib64} \]
   - Modify line 37 as follows:

3. Press Esc, type :wq!, and press Enter to save the file and exit.

Step 7 Run the following command to configure the environment variables of the mathlib library:

\[
\text{export LD_LIBRARY_PATH=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc:SLD_LIBRARY_PATH}
\]

Step 8 Run the following command to compile and install Wannie90:

\[
\text{make -j 16}
\]
\[
\text{make install}
\]

**NOTE**

After the compilation and installation are complete, two executable files \texttt{postw90.x} and \texttt{wannier90.x} are generated.

----End

### 3.25.7 Running and Verification

#### Procedure

**Step 1** Run the following command to go to the test directory:

\[
\text{cd /path/to/WANNIER90/wannier90-3.1.0/examples/example03}
\]

**Step 2** Run the following command to perform the test:

\[
\text{time -p mpirun -np 128 --allow-run-as-root -mca pml ucx -mca btl ^vader,tcp,openib,uct -x UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 ../-/wannier90.x silicon}
\]

After the command is successfully executed, a \texttt{silicon.wout} file is generated. The following figure shows an example of the command output.

**Step 3** Run the following commands to modify the \texttt{silicon.win} file:

1. Open the \texttt{silicon.win} file.
   \[
   \text{vi silicon.win}
   \]
2. Press `i` to go to the edit mode. Add the following two lines to the end of line 2:

```
restart           = plot
bands_plot        = true
```

3. Press `Esc`, type `.wq!`, and press `Enter` to save the file and exit.

**Step 4** Run the test command of **Step 2** again. After the command is successfully executed, the `silicon_band.gnu` and `silicon_band.dat` files are generated in the current directory.

```
time -p mpirun -np 128 --allow-run-as-root -mca pml ucx -mca btl
^vader,tcp,openib,uct -x UCX_TLS=self,sm --bind-to core --map-by socket --rank-by core -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=8 ../../wannier90.x
silicon
```

**Step 5** Run the following command to use the gnuplot interactive graphing tool:

```
gnuplot
```

The following information is displayed:

```
 load '/path/to/WANNIER90/wannier90-3.1.0/examples/example03/silicon_band.gnu'
```

The following information is displayed:

```
----End
```

### 3.25.8 More Resources

Wannier90 official website:
3.26 WanT 2.6.1 Porting Guide (CentOS 7.6)

3.26.1 Introduction

WanT simulates quantum transport properties (conductance) of nanojunctions using a real space Wannier function description of the Hamiltonian of the system. The core methodology combines state-of-the-art Density Functional Theory (DFT), plane-wave, norm-conserving pseudopotentials calculations with a Green’s functions method based on the Landauer formalism to describe quantum conductance. A crucial step in the calculation is the use of the maximally-localized Wannier function representation to introduce naturally the ground-state electronic structure into the lattice Green’s function approach. The WanT package operates, in principles, as a simple post-processing of any standard electronic structure code. The WanT code is currently interfaced to the codes in the Quantum-ESPRESSO distribution, Abinit, and CRYSTAL09. Interfaces with CP2K and GAMESS are under testing in the development version of the code.

Programming language: Fortran

Brief description: The WanT package operates, in principles, as a simple post-processing of any standard electronic structure code.

Open source license: GPL-2.0

Recommended Software Version

want2.6.1

3.26.2 Environment Requirements

Hardware Requirements

Table 3-126 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 3-127 lists the software requirements.
Table 3-127 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>WanT</td>
<td>2.6.1</td>
<td><a href="https://github.com/QEF/want/archive/want-2.6.1.tar.gz">https://github.com/QEF/want/archive/want-2.6.1.tar.gz</a></td>
</tr>
<tr>
<td>QE</td>
<td>4.6.1</td>
<td><a href="https://github.com/QEF/q-e/archive/refs/tags/qe-6.4.1.tar.gz">https://github.com/QEF/q-e/archive/refs/tags/qe-6.4.1.tar.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 3-128 lists the OS requirements.

Table 3-128 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

3.26.3 Paths for Software Porting

This chapter describes the software installation paths involved in the WanT software porting.

Table 3-129 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/path/to/KPGCC</td>
<td>Installation path of the Kunpeng GCC.</td>
<td>For details, see section &quot;Kunpeng GCC Installation and Operation&quot; in User Guide (Kunpeng GCC).</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HMPI</td>
<td>Installation path of the Hyper MPI.</td>
<td>For details, see section &quot;Source Code Installation&quot; in Hyper MPI User Guide.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/WANT</td>
<td>Installation path of WanT.</td>
<td>The installation paths provided here are only an example. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>4</td>
<td>/path/to/QE</td>
<td>Installation path of QE.</td>
<td></td>
</tr>
</tbody>
</table>

Kunpeng BoostKit for HPC Porting Guide 3 Government HPC
3.26.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configure Process

**Table 3-130 Configure process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the open source environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Setting up the Huawei Kunpeng compiler environment</td>
<td>For details, see section &quot;Installing the Kunpeng GCC&quot; in <em>User Guide (Kunpeng GCC)</em>.</td>
</tr>
<tr>
<td>3</td>
<td>Setting up the Huawei MPI environment</td>
<td>For details, see section &quot;Installation Introduction&quot; in <em>Hyper MPI User Guide</em>.</td>
</tr>
</tbody>
</table>

3.26.4.1 Installing QE

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the Yum command to install the LAPACK and BLAS libraries:

```
yum install lapack* blas* -y
```

**Step 3** Run the following commands to decompress the QE installation package:

```
cd /path/to/QE

tar -xvf q-e-qe-6.4.1.tar.gz
```

**Step 4** Run the following command to go to the directory generated after decompression:

```
cd q-e-qe-6.4.1
```

**Step 5** Run the following command to perform the configuration:

```
./configure F90=gfortran F77=gfortran MPIF90=mpifort MPIF77=mpifort CC=mpicc FCFLAGS="-O3" CFLAGS="-O3" --with-scalapack=yes --build=aarch64-unknown-linux-gnu --prefix=/path/to/QE
```

**Step 6** Run the following command to modify the **PP/src/Makefile** file:

1. Open the **PP/src/Makefile** file.

   `vi PP/src/Makefile`
2. Press i to go to the edit mode.
   - Add the following line below line 49:
     ```c
     pw_export.o
     ```
   - Add the following content to the end of line 66:
     ```c
     pw_export.x
     ```
   - Add four lines below line 221:
     i. In common mode, move the cursor to line 217 and press 4yy to copy lines 217 to 220.
     ii. Move the cursor to line 222 and press p to paste.
     iii. Modify the content of lines 222 and 224, as shown in the following figure:

     ```c
     217 fermi_proj.o : fermi_proj.o libpp.a $(MODULES) ${LIBOBSJS}
     218 $(LD) $(LDFLAGS) -o fermiti
     219   fermi_proj.o libpp.a $(MODULES) ${LIBOBSJS} $(QELIBS)
     220   - ( cd ..../bin ; ln -fs ../PP/src/Q@ .)
     221
     222 pw_export.x : pw_export.o libpp.a $(MODULES) ${LIBOBSJS}
     223 $(LD) $(LDFLAGS) -o fermiti
     224   pw_export.o libpp.a $(MODULES) ${LIBOBSJS} $(QELIBS)
     225   - ( cd ..../bin ; ln -fs ../PP/src/Q@ .)
     226
     ```

     □ NOTE

     If you add lines 222 to 225 in edit mode, the font color of part of the content does not change. That is, the modification does not take effect and the executable file of `pw_export.x` fails to be compiled. Therefore, copy the first four lines and keep them consistent before modifying part of the content.

3. Press Esc, type :wq!, and press Enter to save the file and exit.

   **Step 7** Run the following commands to perform the compilation and installation.

   ```bash
   make -j 16 pwall
   make install
   ----End
   ```

   3.26.5 Obtaining the Source Code

   **Procedure**

   **Step 1** Download the `want-want-2.6.1.tar.gz` installation package.

   Download address: https://github.com/QEF/want/archive/want-2.6.1.tar.gz

   **Step 2** Use an SFTP tool to upload the WanT installation package to the `/path/to/WANT` directory on the server.

   ----End
3.26.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the main program installation directory:

```
cd /path/to/WANT
```

Step 3 Run the following command to decompress the main program installation package:

```
tar -xvf want-want-2.6.1.tar.gz
```

Step 4 Run the following command to go to the directory generated after decompression:

```
cd want-want-2.6.1
```

Step 5 Run the following command to perform the configuration:

```
./configure CC=gcc CFLAGS="-O3 -march=armv8.2-a -mcpu=tsv110" --build=aarch64-unknown-linux-gnu
```

Step 6 Run the following commands to modify the `make.sys` file and link the `mathlib` math library of the optimized Kunpeng GCC version to the main program.

1. Open the `make.sys` file.

```
vi make.sys
```

2. Press `i` to go to the edit mode.

   - Add a newline character (`\`) at the end of line 91.
   - Add line 92 and add the following content:

```
```

Step 7 Run the following command to configure the environment variables of the `mathlib` library:

```
export LD_LIBRARY_PATH=/path/to/KPGCC/gcc-9.3.1-2020.12-aarch64-linux/lib64/libhpc:SLD_LIBRARY_PATH
```

Step 8 Run the following command to perform the compilation and installation:

```
make all
```

After the compilation and installation are complete, the following executable files are generated in the `bin` directory:

```
bandi cmlx_bandi current.x distance(x) mead.x regular x grid.x plot.x salt2xaml x subrpg.wannix write.hxm blow_x condens.x decay.x do.x probplot.x plot.x mdcomm.x salt2xml sumdos unfold.x x%20tfx
```

---End
3.26.7 Running and Verification

Procedure

Step 1  Run the following command to go to the tests directory:

```
cd /path/to/WANT/want-want-2.6.1/tests
```

Step 2  Run the following commands to modify the environment.conf file:

1. Open the environment.conf file.
   ```
   vi environment.conf
   ```
2. Press `i` to go to the edit mode.
   - Modify line 28 as follows:
     ```
     QE_BIN=/path/to/QE/q-eqe-6.4.1/bin
     ```
   - Modify line 32 as follows:
     ```
     TMPDIR=/path/to/WANT/want-want-2.6.1
     ```
   - Modify line 34 as follows:
     ```
     PARA_PREFIX="time -p mpirun -np 121 --allow-run-as-root --mca coll ^hcoll -mca pml ucx -
     mca btl ^vader,tcp,openib,uct -x UCX_TLS=self,sm --bind-to core --map-by socket --rank-by
     core -x UCX_BUILTIN_ALLREDUCE_ALGORITHM=8"
     ```
   - Modify line 35 as follows:
     ```
     OMP_NUM_THREADS=1
     ```
3. Press `Esc`, type `:wq!`, and press `Enter` to save the file and exit.

Step 3  Run the following command to go to the test01 directory:

```
cd test01
```

Step 4  Run the following commands to modify the scf.in, nscf.in, and want.in files:

1. Open the scf.in, nscf.in, and want.in files.
   ```
   vi Input file
   ```
2. Press `i` to go to the edit mode.
   a. Add `diagonalization = 'cg'` to the `&electrons` part of scf.in and nscf.in.
   ```
   &electrons
   mixing_mode = 'plain'
   mixing_beta = 0.7
   conv_thr = 1.0d-8
   diagonalization = 'cg'
   ```
   b. Add `nbnd=16` to the nscf.in file.
   ```
   &system
   ibrav = 2, collcolm(1) =10.599478, nat= 2, ntyp= 1,
   ecutwfc =30.0, nbnd = 16
   ```
   c. In the want.in file, comment out `dftdata_fmt`, that is, delete the exclamation mark (!) before `dftdata_fmt`.
   ```
   &CONTROL
   prefix = 'silicon'
   postfix = '.WanT'
   work_dir = ':-/SCRATCH'
   dftdata_fmt = 'pw export'
   ```
3. Press **Esc**, type **:wq!**, and press **Enter** to save the file and exit.

**Step 5**  Run the following command to execute the test:

```
./run.sh all
```

If the test is executed successfully, the following information is displayed:

```
running SCF calculation... real 2.27
user 140.05
sys 51.27
done
running NSCF calculation... real 68.02
user 7822.20
sys 319.17
done
running EXPORT calculation... done
running DISENTANGLE calculation... real 1.89
user 139.30
sys 13.75
done
running WANNIER calculation... real 95.18
user 11426.93
sys 16.13
done
running BANDS calculation... real 1.06
user 48.02
sys 11.81
done
```

----End

3.26.8 More Resources

WanT official website:

http://www.wannier-transport.org
4.1 Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6)
4.2 SGE 8.1.9 Porting Guide (CentOS 7.6)
4.3 Conda 4.3.16 Porting Guide (CentOS 7.6)
4.4 MiniFE 2.2.0 Porting Guide (CentOS 7.6)
4.5 NPB 3.4 Porting Guide (CentOS 7.6)
4.6 MiniDFT 1.1.1 Porting Guide (CentOS 7.6)
4.7 miniGhost v1.0 Porting Guide (CentOS 7.6)
4.8 BLAST 2.9.0 Porting Guide (CentOS 7.6)
4.9 BWA 0.7.17 Porting Guide (CentOS 7.6)
4.10 CNVnator 0.4.1 Porting Guide (CentOS 7.6)
4.11 Cufflinks 2.2.1 Porting Guide (CentOS 7.6)
4.12 GATK 4.0.0.0 Porting Guide (CentOS 7.6)
4.13 HISAT2 2.1.0 Porting Guide (CentOS 7.6)
4.14 STAR 2.7.1a Porting Guide (CentOS 7.6)
4.15 Tophat2 2.1.1 Porting Guide (CentOS 7.6)
4.16 Bowtie2 2.4.1 Porting Guide (CentOS 7.6)
4.17 FastQC 0.11.9 Porting Guide (CentOS 7.6)
4.18 Beagle 5.1 Porting Guide (CentOS 7.6)
4.19 Bowtie 1.2.3 Porting Guide (CentOS 7.6)
4.20 pblat 2.1 Porting Guide (CentOS 7.6)
4.21 SnpEff V4.3 Porting Guide (CentOS 7.6)
4.22 Velvet 1.2.10 Porting Guide (CentOS 7.6)
4.23 XHMM Porting Guide (CentOS 7.6)
4.24 pysam 0.15.4 Porting Guide (CentOS 7.6)
4.25 Abyss 2.2.4 Porting Guide (CentOS 7.6)
4.26 Chaste 2019.1 Porting Guide (CentOS 7.6)
4.27 COPASI 4.27.217 Porting Guide (CentOS 7.6)
4.28 FASTA 36.3.8 Porting Guide (CentOS 7.6)
4.29 gmap 2015.9.21 Porting Guide (CentOS 7.6)
4.30 VarScan 2.4.2 Porting Guide (CentOS 7.6)
4.31 Minimap2 Porting Guide (CentOS 7.6)
4.32 bedtools 2.29.2 Porting Guide (CentOS 7.6)
4.33 BEAST2 2.6.3 Porting Guide (CentOS 7.6)
4.34 bcftools 1.10.2 Porting Guide (CentOS 7.6)
4.35 picard 2.23.3 Porting Guide (CentOS 7.6)
4.36 SOAPdenovo r241 Porting Guide (CentOS 7.6)
4.37 Blastz 2004-12-27 Porting Guide (CentOS 7.6)
4.38 CAFE 5.0b2 Porting Guide (CentOS 7.6)
4.39 fastp Porting Guide (CentOS 7.6)
4.40 GCE 1.0.2 Porting Guide (CentOS 7.6)
4.41 GEMMA 0.98.1 Porting Guide (CentOS 7.6)
4.42 kmergenie 1.7051 Porting Guide (CentOS 7.6)
4.43 LASTZ 1.04.03 Porting Guide (CentOS 7.6)
4.44 MAFFT Porting Guide (CentOS 7.6)
4.45 megahit 1.2.9 Porting Guide (CentOS 7.6)
4.46 PASA 2.4.1 Porting Guide (CentOS 7.6)
4.47 Augustus 3.3.3 Porting Guide (CentOS 7.6)
4.48 Delly 0.8.5 Porting Guide (CentOS 7.6)
4.49 FastTree 2.1.11 Porting Guide (CentOS 7.6)
4.50 Jellyfish 2.2.10 Porting Guide (Centos 7.6)
4.51 PLINK 1.9 Porting Guide (CentOS 7.6)
4.52 SNAP 2013-11-29 Porting Guide (CentOS 7.6)
4.53 kmersGWAS 0.2 Porting Guide (CentOS 7.6)
4.54 PAML 4.9j Porting Guide (CentOS 7.6)
4.55 Diamond 2.0.4 Porting Guide (CentOS 7.6)
4.56 HISAT 0.1.6 Porting Guide (CentOS 7.6)
4.1 Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6)

4.1.1 Introduction

Hadoop is an Apache open source framework written in Java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

For more information about Hadoop, visit the official Hadoop website.

Language: Java

Brief description: open source code framework

Open source protocol: Apache 2.0

4.1.2 Environment Requirements

Hardware Requirements

Table 4-1 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-2 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop</td>
<td>3.1.2</td>
<td><a href="http://mirror.bit.edu.cn/apache/hadoop/common/stable/">http://mirror.bit.edu.cn/apache/hadoop/common/stable/</a></td>
</tr>
<tr>
<td>Spark</td>
<td>2.4.4</td>
<td><a href="https://spark.apache.org/releases/spark-release-2-4-4.html">https://spark.apache.org/releases/spark-release-2-4-4.html</a></td>
</tr>
<tr>
<td>Item</td>
<td>Version</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

### OS Requirements

Table 4-3 lists the OS requirements.

**Table 4-3 OS requirements**

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.1.3 Planning the Paths for Software Porting

Table 4-4 lists the software installation paths involved in the Hadoop software porting.

**Table 4-4 Paths for software porting**

<table>
<thead>
<tr>
<th>No.</th>
<th>Software installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HADOOP</td>
<td>Installation path of Hadoop</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>
4.1.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Procedure

Table 4-5 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Install the Java environment.</td>
<td>See 4.1.4.1 Installing the Java Environment.</td>
</tr>
</tbody>
</table>

4.1.4.1 Installing the Java Environment

Procedure

Step 1 Obtain the Java package jdk-8u152-linux-arm64-vfp-hflt.tar.gz. For details, see the software requirements.

Step 2 Use PuTTY to log in to the server as the root user.

Step 3 Run the following commands to install the Java environment:

```bash
mkdir /path/to/HADOOP
cd path/to/HADOOP
tar -zxvf jdk-8u152-linux-arm64-vfp-hflt.tar.gz
```

**NOTE**

Differences between JRE and JDK:

- The Java Runtime Environment (JRE) is the environment required for Java running.
- The Java Development Kit (JDK) includes the JRE and the tools and class libraries required for Java development.

Step 4 Configure environment variables and modify the /root/.bashrc file.

1. `vi /root/.bashrc`
2. Press `i` to enter the editing mode and add the following path to the end of the `/root/.bashrc` file:
   ```bash
   export JAVA_HOME=/path/to/HADOOP/jdk1.8.0_171
   export JRE_HOME=$JAVA_HOME/jre
   export PATH=$JAVA_HOME/bin:$PATH
   export CLASSPATH=$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
   ```
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

----End
4.1.5 Obtaining the Source Code

Procedure

Step 1  Download the Hadoop installation package hadoop-3.1.2.tar.gz.
        Download address: http://mirror.bit.edu.cn/apache/hadoop/common/stable/

Step 2  Use SFTP to upload the Hadoop installation package to the /path/to/HADOOP directory on the server.

----End

4.1.6 Compiling and Installing Hadoop and Spark

4.1.6.1 Configuring Hadoop

Hadoop can run on a single node in pseudo-distributed mode. The Hadoop process runs as a separated Java process. The node functions as both the NameNode and DataNode and reads files in the HDFS.

The configuration file is stored in hadoop-3.1.2/etc/hadoop. For a pseudo-distributed cluster, you need to modify the core-site.xml and hdfs-site.xml configuration files. The Hadoop configuration file is in XML format. Each configuration is implemented by declaring the name and value of property.

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to switch to the directory, in which the Hadoop installation package is stored:
        cd path/to/HADOOP

Step 3  Run the following command to decompress the Hadoop installation package:
        tar -xvf hadoop-3.1.2.tar.gz

Step 4  Run the following command to switch to the directory generated after the package is decompressed:
        cd hadoop-3.1.2

Step 5  Run the following commands to create the following four folders in the hadoop-3.1.2 folder as the HDFS file paths:
        mkdir hdfs
        mkdir hdfs/tmp
        mkdir hdfs/name
        mkdir hdfs/data

Step 6  Run the following command to go to the directory in which the configuration file is stored:
**cd etc/hadoop/**

**Step 7** Run the following command to modify the `core-site.xml` file:

1. `vi core-site.xml`
2. Press `i` to enter the editing mode and edit the `core-site.xml` file.

   **Before the modification:**
   ```xml
   <configuration>
   </configuration>
   ```

   **After the modification:**
   ```xml
   <configuration>
   <property>
     <name>hadoop.tmp.dir</name>
     <value>file:/path/to/HADOOP/hadoop-3.1.2/hdfs/tmp</value>
     <description>Abase for other temporary directories.</description>
   </property>
   <property>
     <name>io.file.buffer.size</name>
     <value>4096</value>
   </property>
   <property>
     <name>fs.defaultFS</name>
     <value>hdfs://armnode2:9000</value>
   </property>
   <property>
     <name>fs.trash.interval</name>
     <value>1440</value>
   </property>
   </configuration>
   ```

   **NOTE**

   `armnode2` indicates the hostname of the installation environment. You can set this parameter based on the actual situation. You can run the `hostname` command to query the hostname of the installation environment.

3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

**Step 8** Run the following command to modify the `hdfs-site.xml` configuration file:

1. `vi hdfs-site.xml`
2. Press `i` to enter the editing mode and edit the `hdfs-site.xml` file.

   **Before the modification:**
   ```xml
   <configuration>
   </configuration>
   ```

   **After the modification:**
   ```xml
   <configuration>
   <property>
     <name>dfs.namenode.http-address</name>
     <value>armnode2:50070</value>
   </property>
   <property>
     <name>dfs.replication</name>
     <value>1</value>
   </property>
   <property>
     <name>dfs.namenode.name.dir</name>
     <value>file:/path/to/HADOOP/hadoop-3.1.2/hdfs/name</value>
   </property>
   <property>
     <name>dfs.datanode.data.dir</name>
     <value>file:/path/to/HADOOP/hadoop-3.1.2/hdfs/data</value>
   </property>
   </configuration>
   ```
<property>
  <name>dfs.webmnt.enabled</name>
  <value>true</value>
</property>

<property>
  <name>dfs.permissions</name>
  <value>false</value>
</property>
</configuration>

NOTE

armnode2 indicates the hostname of the installation environment. You can set this parameter based on the actual situation. You can run the hostname command to query the hostname of the installation environment.

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 9** Run the following commands to modify the workers configuration file:

1. vi workers
2. Press i to enter the editing mode and edit the workers file to add the hostname of the installation environment. armnode2
3. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 10** Run the following commands to create the master and slaves configuration files:

```bash
cd /path/to/HADOOP/hadoop-3.1.2/etc/hadoop
cp workers master
cp workers slaves
```

----End

### 4.1.6.2 Configuring Spark

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the Spark installation package:

```bash
tar -xvf spark-2.4.4-bin-hadoop2.7.tgz
```

**Step 3** Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd /path/to/HADOOP/spark-2.4.4-bin-hadoop2.7/conf
```

**Step 4** Run the following commands to modify the spark-defaults.conf configuration file:

1. mv spark-defaults.conf.template spark-defaults.conf
2. vi spark-defaults.conf
3. Press i to enter the editing mode and add the following content to the spark-defaults.conf file:

   ```
   spark.master                     spark://armnode2:7077
   spark.scheduler.mode            FAIR
   spark.eventLog.enabled          true
   ```
spark.eventLog.dir  hdfs://armnode2:9000/sparklog
spark.shuffle.consolidateFiles  true
spark.shuffle.manager  SORT
spark.sql.hive.convertMetastoreOrc false

- **NOTE**

armnode2 indicates the hostname of the installation environment. You can set this parameter based on the actual situation. You can run the `hostname` command to query the hostname of the installation environment.

4. Press Esc, enter `.wq!`, and press Enter to save the file and exit.

**Step 5** Run the following commands to modify the `slaves` configuration file:

1. `mv slaves.template slaves`
2. `vi slaves`
3. Press i to enter the editing mode and edit the `slaves` file to add the hostname of the installation environment.
   armnode2
4. Press Esc, enter `.wq!`, and press Enter to save the file and exit.

**Step 6** Run the following commands to modify the `spark-env.sh` configuration file:

1. `mv spark-env.sh.template spark-env.sh`
2. `vi spark-env.sh`
3. Press i to enter the editing mode and add the following content to the `spark-env.sh` file:
   ```
   export JAVA_HOME=/path/to/HADOOP/jdk1.8.0_171
   export SCALA_HOME=/path/to/HADOOP/scala-2.12.4
   export SPARK_HOME=/path/to/HADOOP/spark-2.4.4-bin-hadoop2.7
   export SPARK_MASTER_IP=armnode2
   export HADOOP_HOME=/path/to/HADOOP/hadoop-3.1.2
   export HADOOP_CONF_DIR=/path/to/HADOOP/hadoop-3.1.2/etc/hadoop
   export SPARK_DIST_CLASSPATH=$(/path/to/HADOOP/hadoop-3.1.2/bin/hadoop classpath)
   export SPARK_DRIVER_MEMORY=30g
   export SPARK_MASTER_INSTANCES=10
   export SPARK_MASTER_CORES=16
   export SPARK_MASTER_MEMORY=20g
   export SPARK_EXECUTOR_MEMORY=10g
   export SPARK_LOCAL_DIRS=/path/to/HADOOP/hadoop-3.1.2/hdfs/spark/tmp
   export SPARK_WORKER_DIR=/path/to/HADOOP/hadoop-3.1.2/hdfs/spark/work
   ```

- **NOTE**

armnode2 indicates the hostname of the installation environment. You can set this parameter based on the actual situation. You can run the `hostname` command to query the hostname of the installation environment.

4. Press Esc, enter `.wq!`, and press Enter to save the file and exit.

**Step 7** Run the following command to go to the directory in which the configuration file is stored:

```
cd /path/to/HADOOP
```

**Step 8** Run the following command to configure environment variables:

1. `vi env.sh`
2. Press i to enter the editing mode, create an environment variable file, and add the following content to the file:
   ```
   export JAVA_HOME=/path/to/HADOOP/jdk1.8.0_171
   export JRE_HOME=$JAVA_HOME/jre
   ```
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
export HADOOP_HOME=/path/to/HADOOP/hadoop-3.1.2
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
export HDFS_DATANODE_USER=root
export HDFS_NAMENODE_USER=root
export HDFS_SECONDARYNAMENODE_USER=root
export YARN_RESOURCEMANAGER_USER=root
export YARN_NODEMANAGER_USER=root
export SCALA_HOME=/path/to/HADOOP/scala-2.12.4
export PATH=$SCALA_HOME/bin:$PATH
export SPARK_HOME=/path/to/HADOOP/spark-2.4.4-bin-hadoop2.7
export PATH=$SPARK_HOME/bin:$PATH

NOTE

Hadoop configuration file description:
The running mode of Hadoop is determined by the configuration file read when
Hadoop is running. Therefore, if you need to switch from the pseudo-distributed mode
to the non-distributed mode, you need to delete the configuration items in the core-
site.xml file.

In addition, pseudo-distributed Hadoop can run after fs.defaultFS and dfs.replication
are configured (as described in the official tutorial). However, if hadoop.tmp.dir is not
configured, the default temporary directory is /tmp/hadoop-hadoop, the directory may
be deleted by the system during the restart. As a result, you must run the format
command again. Therefore, you need to specify dfs.namenode.name.dir and
dfs.datanode.data.dir. Otherwise, errors may occur in the following steps.

3. Press Esc, enter .wq!, and press Enter to save the file and exit.
4. Run the following command to make the environment variables take effect:
   ```bash
   source env.sh
   ```

**Step 9** After the configuration is complete, format the NameNode.

```bash
hdfs namenode -format
```

If the following information is displayed, the process ends:

```
/root/GATK/hadoop-3.1.2/hdfs/name has been successfully formatted
```

**Step 10** Run the following commands to start the NameNode and DataNode daemon
processes:

```bash
start-dfs.sh
jps
```

Run the jps command to check whether the NameNode, DataNode, and
SecondaryNameNode processes are successfully started. If they are successfully
started, the processes shown in Figure 4-1 are displayed. If the
SecondaryNameNode process is not started, run the sbin/stop-dfs.sh command to
stop the process and try again. If NameNode or DataNode is not started, the
configuration fails. Check the previous steps or view the startup logs to locate the
fault.

**Figure 4-1** Example

```
[jroot@armnode2 hadoop-3.1.2]# jps
16681 NameNode
16295 DataNode
16666 SecondaryNameNode
28926 Jps
```
Step 11 Run the following command to go to the Spark directory:

```
    cd /path/to/HADOOP/spark-2.4.4-bin-hadoop2.7/sbin
```

Step 12 Run the following command to start the Spark process:

```
./start-all.sh
```

Run the `jps` command to check whether the Spark process is successfully started. If the Spark process is successfully started, multiple Worker processes are displayed, as shown in Figure 4-2.

![jps Command Output Example](image)

Figure 4-2 Example

----End

4.1.7 Running and Verifying Hadoop and Spark

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to create a user directory in the HDFS:

```
    cd path/to/HADOOP

    ./bin/hdfs dfs -mkdir -p /user/hadoop
```

**NOTE**

The directories and files created by HDFS DFS operations can be identified only in Hadoop mode. They cannot be viewed by running the local `ls` command.

Step 3 Run the following commands to create the input directory and copy the files in etc/hadoop to the input directory:

```
    hdfs dfs -mkdir -p input

    hdfs dfs -put ./etc/hadoop/*.xml input
```
Step 4: Run the following command to view the file list:

```bash
hdfs dfs -ls input
```

Step 5: Run the following command to stop Hadoop:

```bash
stop-dfs.sh
```

--- End

4.1.8 Troubleshooting

**Problem 1: An Error Is Reported When hdfs namenode -format Is Run**

**Symptom**

The error message "Error: JAVA_HOME is not set and could not be found" is displayed when `hdfs namenode -format` is run.

**Possible Causes**

The `JAVA_HOME` environment variable is incorrectly set.

**Procedure**

Set the `JAVA_HOME` variable by referring to Configuring the Compilation Environment.

If `JAVA_HOME` is correctly set, modify the `./etc/hadoop/hadoop-env.sh` file.

1. `vi ./etc/hadoop/hadoop-env.sh`
2. Press `i` to enter the editing mode and add the following environment variable to the environment variable file:
   ```bash
   export JAVA_HOME=JDK installation path
   ```
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

**Problem 2: An Error Is Reported When hdfs namenode -format Is Run**

**Symptom**

The error message "Permission denied (publickey, gssapi-keyex, gssapi-with-mic, password)" is displayed when `hdfs namenode -format` is run.

**Possible Causes**

The password-free function of the local host is faulty.

**Procedure**

Run the `ssh-keygen -t rsa` command to enable password-free function again.

**Problem 3: "Could not resolve hostname" Displayed When Hadoop Is Started**

**Symptom**

The error message "ssh: Could not resolve hostname xxx" is displayed when Hadoop is started.
Procedure

You can set the Hadoop environment variable to resolve this problem.

**Step 1** Press Ctrl+C to stop the startup.

**Step 2** Add the following two lines to ~/.bashrc:

1. vi ~/.bashrc
2. Press i to enter the editing mode and add the following commands to ~/.bashrc:
   
   ```bash
   export HADOOP_HOME=/usr/local/hadoop
   export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
   ```

   **NOTE**

   The setting process is the same as that of the JAVA_HOME variable. In the variable, HADOOP_HOME indicates the Hadoop installation directory.

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 3** Run the following command for the setting to take effect:

```
source ~/.bashrc
```

**Step 4** Run the following command to start Hadoop:

```
start-dfs.sh
```

----End

### 4.1.9 More Information

Official Hadoop website:


### 4.2 SGE 8.1.9 Porting Guide (CentOS 7.6)

#### 4.2.1 Introduction

Sun Grid Engine (SGE) is a grid-based resource management software developed by Sun. It receives jobs submitted by users and uses resource management policies to schedule jobs to run on appropriate systems in the grid. Users can submit batch, interactive, and parallel jobs to the grid. It is an advanced resource management tool used to manage different types of distributed computing environments.

The SGE software consists of three modules: sge_qmaster, sge_schedd, and sge_execd. The servers in the cluster are classified into the master hosts, administration hosts, execution hosts, and submission hosts, depending on the module running on them.

For more information, visit the official SGE website.

Programming language: C/Java

Brief description: scheduler software based on grid resource management
Recommended Version

SGE 8.1.9

4.2.2 Environment Requirements

Hardware Requirements

Table 4-6 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-7 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGE</td>
<td>8.1.9</td>
<td><a href="https://arc.liv.ac.uk/downloads/SGE/releases/8.1.9/">https://arc.liv.ac.uk/downloads/SGE/releases/8.1.9/</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-8 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
### Table 4-9 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/path/to/SGE</td>
<td>Directory in which the SGE source code is compiled</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/SGE_ROOT</td>
<td>Installation path of SGE</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4-10 SGE host description

<table>
<thead>
<tr>
<th>No.</th>
<th>Host</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Master host</td>
<td>The master host is central for the overall cluster activities. It runs the sge_qmaster daemon and the sge_schedd daemon. These daemons control grid engine system components, such as queues and jobs. The daemons maintain the tables related to component status, user access permissions, and other tasks. By default, the master host also manages the administration hosts and submission hosts. In this document, the node name of the master host is <code>armnode2</code>.</td>
</tr>
<tr>
<td>2</td>
<td>Administration host</td>
<td>The administration host runs any type of management operations in the grid engine system.</td>
</tr>
<tr>
<td>3</td>
<td>Submission host</td>
<td>Users can submit and control batch processing jobs on the submission hosts. Specifically, after logging in to a submission host, a user can use the <code>qsub</code> command to submit a job, and use the <code>qstat</code> command to monitor the job status.</td>
</tr>
<tr>
<td>4</td>
<td>Execution host</td>
<td>The execution hosts run jobs. Therefore, they host queue instances. The execution hosts run the sge_execd daemon. In this document, the node name of the execution host is <code>armnode4</code>.</td>
</tr>
</tbody>
</table>

### 4.2.4 Configuring the Compilation Environment

**Step 1** Use PuTTY to log in to the server as the `root` user.

**Step 2** Run the following command to install dependencies:

```
yum install -y java-1.8.0-openjdk* ant hwloc-devel openssl-devel libdb-devel pam-devel junit* javacc* libXt-devel motif-devel ncurses-devel
```
In this example, the CentOS 7.6 image is used as the local YUM source. You can change the YUM source in the `/etc/yum.repos.d` directory as required.

**Step 3** Load the environment variables of Java 1.8.0 that has been installed.

```bash
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.181-7.b13.el7.aarch64
export PATH=$JAVA_HOME/bin:$PATH
```

**Step 4** Create a soft link for the JRE library of arm64.

```bash
cd /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.181-7.b13.el7.aarch64/jre/lib
ln -s aarch64 arm64
```

----End

### 4.2.5 Obtaining the Source Code

**Step 1** Download the SGE source code package `sge-8.1.9.tar.gz`.

URL: [https://arc.liv.ac.uk/downloads/SGE/releases/8.1.9/](https://arc.liv.ac.uk/downloads/SGE/releases/8.1.9/)

**Step 2** Use SFTP to upload the SGE source code package to the `/path/to/SGE` directory on the server.

----End

### 4.2.6 Compiling and Installing SGE

#### 4.2.6.1 Compiling SGE

**Step 1** Use PuTTY to log in to the SGE master host as the **root** user.

**Step 2** Run the following commands on PuTTY to decompress the SGE installation package:

```bash
cd /path/to/SGE
tar -xvf sge-8.1.9.tar.gz
cd sge-8.1.9/source
```

**Step 3** Run the `bootstrap.sh` script.

```bash
sh scripts/bootstrap.sh
```

**Step 4** Modify the `aimk` file.

1. Run the `vi aimk` command.
2. Press `I` to enter the insert mode, and add `"-L/usr/lib64 -lcurses"` to the end of `-lpthread` in lines 330, 1852, and 2664 in the `aimk` file.

   330: `set LIBS = "-lm -lpthread -L/usr/lib64 -lcurses"
   1852: `set LIBS = "-lsocket -lnsl -lm -lpthread -L/usr/lib64 -lcurses -lthread -lkstat"
   2664: `set SGE_LIBS = "-lsge -lpthread -L/usr/lib64 -lcurses"

3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.
Step 5 Set the environment variable SGE_ROOT.

```shell
export SGE_ROOT=/path/to/SGE_ROOT
```

Step 6 Run the following commands to compile the software:

```shell
./aimk -parallel 32
echo y | ./scripts/distinst -all -local -noexit
```

----End

4.2.6.2 Installing SGE on the Master Host

Step 1 Run the following command to install the NIS package:

```shell
yum install ypserv -y
```

Step 2 Configure the NIS domain.

```shell
nisdomainname huawei
```

□ NOTE

In the command, *huawei* indicates the domain name of the cluster. Replace it with the actual domain name.

Step 3 Modify the `/etc/sysconfig/network` configuration file.

1. `vi /etc/sysconfig/network`
2. Press `i` to enter the insert mode and enter the following information:

```
HOSTNAME=Name of the master host
NETWORKING=yes
NISDOMAIN=huawei
```

□ NOTE

*HOSTNAME* indicates the name of the master node.

3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

Step 4 Add a user, for example, user test.

```shell
useradd -d /path/test test
```

□ NOTE

In this example, `/path` is a shared directory. All nodes in the cluster are mounted to the `/path` directory. Set the path based on actual situation.

Step 5 Change the password of the test user, for example, to Huawei12#$.

```shell
echo "Huawei12#$" | passwd --stdin test
```

Step 6 Restart the ypserv service to update the database.

```shell
systemctl start ypserv
systemctl enable ypserv
cd /var/yp
make
```
Step 7 Run the following commands to start the installation:

```bash
cd /path/to/SGE_ROOT
./install_qmaster
```

Step 8 On the **Welcome to the Grid Engine installation** screen, press **Enter**.

![Welcome to the Grid Engine installation](image1)

Step 9 On the **Choosing Grid Engine admin user account** screen, enter `n` and press **Enter** twice.

![Choosing Grid Engine admin user account](image2)

Step 10 On the **Checking $SGE_ROOT directory** screen, use the path planned and press **Enter** twice.

![Checking $SGE_ROOT directory](image3)
Step 11 On the Grid Engine TCP/IP communication service screen, enter 2 and press Enter.


Step 13 On the Grid Engine TCP/IP communication service screen, enter 2 and press Enter.

Step 14 On the Grid Engine TCP/IP communication service screen, press Enter.
Step 15 On the *Grid Engine cells* screen, use the default cell name and press **Enter** twice.

Step 16 On the *Unique Cluster name* screen, enter the cluster name and press **Enter** twice.

Step 17 On the *Grid Engine qmaster spool directory* screen, specify the spool directory (the default directory is recommended) and press **Enter** twice.
Step 18 On the **Windows Execution Host Support** screen, enter `n` and press **Enter**.

Step 19 On the **Verifying and setting file permissions** screen, enter `y` and press **Enter** twice.

Step 20 On the **Select default Grid Engine hostname resolving method** screen, enter `y` and press **Enter** twice.

Step 21 On the **Grid Engine JMX MBean server** screen, enter `n` and press **Enter** twice.
Step 22 On the **Setup spooling** screen, enter `classic` and press **Enter**.

Step 23 On the **Grid Engine group id range** screen, use the default settings and press **Enter** twice.

Step 24 On the **Grid Engine cluster configuration** screen, use the default settings and press **Enter**.
Step 25  On the Grid Engine cluster configuration (continued) screen, use the default settings and press Enter.

Step 26  When "The following parameters for the cluster configuration were configured" is displayed, enter n and press Enter.

Step 27  On the Creating local configuration screen, press Enter.

Step 28  On the qmaster startup script screen, enter y and press Enter twice.

Step 29  On the Grid Engine qmaster startup screen, press Enter.
Step 30 On the Adding Grid Engine hosts screen, enter n and press Enter.

Adding Grid Engine hosts

Please now add the list of hosts where you will later install your execution daemons. These hosts will be also added as valid submit hosts.

Please enter a blank-separated list of your execution hosts. You may press <RETURN> if the line is getting too long. Once you are finished simply press <RETURN> without entering a name.

You also may prepare a file with the hostnames of the machines where you plan to install Grid Engine. This may be convenient if you are installing Grid Engine on many hosts.

Do you want to use a file which contains the list of hosts (y/n) [n] >> n

Step 31 On the Adding admin and submit hosts screen, press Enter twice.

Adding admin and submit hosts

Please enter a blank-separated list of hosts.

Stop by entering <RETURN>. You may repeat this step until you are entering an empty list. You will see messages from Grid Engine when the hosts are added.

Host(s):
Finished adding hosts. Hit <RETURN> to continue >>

Step 32 On the Shadow host screen, enter n and press Enter.

If you want to use a shadow host, it is recommended to add this host to the list of administrative hosts.

If you are not sure, it is also possible to add or remove hosts after the installation with <qconf -ah hostname> for adding and <qconf -dh hostname> for removing this host

Attention: This is not the shadow host installation procedure.
You still have to install the shadow host separately

Do you want to add your shadow host(s) now? (y/n) [n] >> n

Step 33 On the Creating the default queue and hostgroup screen, press Enter.

Creating the default <all.q> queue and <allhosts> hostgroup

root@armnode2 added "@allhosts" to host group list
root@armnode2 added "all.q" to cluster queue list

Hit <RETURN> to continue >>

Step 34 On the Scheduler Tuning screen, enter 1 and press Enter. Then, enter y and press Enter.
Step 35 On the **Using Grid Engine** screen, press **Enter**.

Step 36 On the **Grid Engine messages** screen, enter **n** and press **Enter**.

Step 37 When "Your Grid Engine qmaster installation is now completed" is displayed, press **Enter** to complete the installation.
Step 38 Load the SGE environment variables.

```
source /path/to/SGE_ROOT/default/common/settings.sh
```

Step 39 Add the SGE environment variables to the .bashrc file for the environment variables to take effect permanently.

```
echo "source /path/to/SGE_ROOT/default/common/settings.sh" >> /root/.bashrc
```

Step 40 Add the master host as the submission host.

```
qconf -as armnode2
```

Step 41 Add the master host as the administration host.

```
qconf -ah armnode4
```

---End

### 4.2.6.3 Installing SGE on the Execution Host

**Step 1** On the execution host, run the following command to install the NIS client installation package ypbind:

```
yum install ypbind -y
```

**NOTE**

In this example, the CentOS 7.6 image is used as the local YUM source. You can change the YUM source in the `/etc/yum.repos.d` directory as required.

**Step 2** Modify the `/etc/sysconfig/network` file on the execution host.

1. Run the `vi /etc/sysconfig/network` command.
2. Press `i` to enter the insert mode and enter the following information:

```
HOSTNAME=Name of the execution host
NETWORKING=yes
NISDOMAIN=huawei
```

**NOTE**

HOSTNAME specifies the name of the execution host, and NISDOMAIN specifies the domain name of the cluster. Set them based on actual situation.
3. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 3** Modify the `/etc/yp.conf` file on the execution host.

1. Run the `vi /etc/yp.conf` command.
2. Press i to enter the insert mode and add the following information to the end of the file:

   ```
   domain huawei server 192.168.40.50
   ```

   **NOTE**
   
   In this example, 192.168.40.50 indicates the IP address of the master host. Replace it with the actual IP address.

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 4** Modify the `/etc/nsswitch.conf` file.

1. vi `/etc/nsswitch.conf`
2. Press i to enter the insert mode and enter the following information:

   ```
   passwd: files nis
   shadow: files nis
   group: files nis
   hosts: files nis
dns
   ```

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 5** Start the ypbind service.

```
systemctl start ypbind
systemctl enable ypbind
```

**Step 6** Install hwloc.

```
yum install hwloc* -y
```

**NOTE**

In this example, the CentOS 7.6 image is used as the local YUM source. You can change the YUM source in the `/etc/yum.repos.d` directory as required.

**Step 7** Set the environment variable `SGE_ROOT`.

```
export SGE_ROOT=/path/to/SGE_ROOT
```

**Step 8** Load the SGE environment variables.

```
source /path/to/SGE_ROOT/default/common/settings.sh
```

**Step 9** Add the SGE environment variables to the `.bashrc` file for the environment variables to take effect permanently.

```
echo "source /path/to/SGE_ROOT/default/common/settings.sh" >> /root/.bashrc
```

**Step 10** Start the execd process on the execution host.

```
/path/to/SGE_ROOT/default/common/sgeexecd
```

**Step 11** Enable the execd process to automatically start as the system boots.

```
echo "/path/to/SGE_ROOT/default/common/sgeexecd" >> /etc/rc.d/rc.local
```
4.2.7 Running and Verifying SGE

Procedure

**Step 1** Use PuTTY to log in to the SGE master host as the root user.

**Step 2** Run the following command to add the execution host:

```
qconf -as armnode4
```

**Step 3** Query the default host name.

```
qconf -shgrpl
```

**Step 4** Modify the host group information and add the execution host to the host group.

1. `qconf -mhgrp @allhosts`
2. Press `i` to enter the insert mode and enter the following information:
   - `group_name @allhosts`
   - `hostlist armnode4`
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

**Step 5** Switch to a non-test user.

```
su - test
```

**Step 6** Load the SGE environment variables as the test user.

```
source /path/to/SGE_ROOT/default/common/settings.sh
```

**Step 7** Add the SGE environment variables to the `.bashrc` file of the test user for the environment variables to take effect permanently.

```
echo "source /path/to/SGE_ROOT/default/common/settings.sh" >> /path/test/.bashrc
```

**Step 8** Create an execution script `run.sh`.

1. Run the `vi run.sh` command.
2. Press `i` to enter the insert mode and add the following information:
   - `#!/bin/bash`
   - `#$ -S /bin/bash`
   - `#S -S /bin/bash`
   - `nodeinfo=`hostname`
   - `echo "This is the SGE test from $nodeinfo" >> sge-test.log`
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

**Step 9** Submit a job.

```
qsub -V -cwd -o stdout.txt -e stderr.txt run.sh
```

Table 4-11 describes the command parameters.
Table 4-11 Common parameters in the `qsub` command

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-V</td>
<td>Exports the environment variables in the current shell to the job to be submitted.</td>
</tr>
<tr>
<td>-cwd</td>
<td>Runs the program in the current working directory. By default, it is the home directory on the compute node of the current user.</td>
</tr>
<tr>
<td>-o</td>
<td>Adds the standard output to the end of the specified file. The default file name is <code>$job_name.o$job_id</code>.</td>
</tr>
<tr>
<td>-e</td>
<td>Adds the standard error output to the end of the specified file. The default file name is <code>$job_name.e$job_id</code>.</td>
</tr>
<tr>
<td>-q</td>
<td>Specifies the queue to be delivered. If this parameter is not specified, the system searches for the queue with the permission and minimum load to execute the job.</td>
</tr>
<tr>
<td>-S</td>
<td>Specifies the software that runs the commands in <code>run.sh</code>. The default value is <code>tcsh</code>. You are advised to use bash. Set this parameter to <code>/bin/bash</code>, or add &quot;#$ -S /bin/bash&quot; at the beginning of the <code>run.sh</code> file. If the parameter is not set to bash, the output contains &quot;Warning: no access to tty (Bad file descriptor).&quot;</td>
</tr>
<tr>
<td>-hold_jid</td>
<td>Specifies the jobs to be executed before the current job. A comma (,) is used to separate multiple job_ids.</td>
</tr>
<tr>
<td>-N</td>
<td>Sets the job name. The default job name is the input file name of qsub.</td>
</tr>
<tr>
<td>-p</td>
<td>Sets the job priority. The value ranges from -1023 to 1024. A larger value indicates a higher priority. However, a higher permission is required to set this parameter to a positive number. Common users cannot set this parameter to a positive number.</td>
</tr>
<tr>
<td>-j y/n</td>
<td>Specifies whether to merge the standard output and standard error output streams into the <code>-o</code> parameter result.</td>
</tr>
<tr>
<td>-pe</td>
<td>Sets parallel environment (PE) information.</td>
</tr>
</tbody>
</table>

Step 10 Manage jobs. Table 4-12 describes the commands and parameter for job management.
### Table 4-12 Commands and parameter for job management

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qstat -f</td>
<td>Queries all jobs submitted by the current user on the current node. The job status can be:</td>
</tr>
<tr>
<td></td>
<td>● <strong>qw</strong>: waiting state. A job is in <strong>qw</strong> state after being submitted. The job runs immediately once computing resources are available.</td>
</tr>
<tr>
<td></td>
<td>● <strong>hqw</strong>: pending state. A job in this state starts to run only after the previous jobs are executed. The job submitted by using qsub with <strong>-hold_jid</strong> specified is in this state.</td>
</tr>
<tr>
<td></td>
<td>● <strong>Eqw</strong>: job waiting in error state.</td>
</tr>
<tr>
<td></td>
<td>● <strong>r</strong>: running state.</td>
</tr>
<tr>
<td></td>
<td>● <strong>s</strong>: job suspended temporarily because resources are used by the job with a higher priority.</td>
</tr>
<tr>
<td></td>
<td>● <strong>dr</strong>: job deleted after a node exits unexpectedly. The job disappears only after the node is restarted.</td>
</tr>
<tr>
<td>qstat jobID</td>
<td>Queries information by job ID.</td>
</tr>
<tr>
<td>qstat -u user</td>
<td>Queries information by user.</td>
</tr>
<tr>
<td>qdel jobID</td>
<td>Deletes a job.</td>
</tr>
</tbody>
</table>

### 4.2.8 SGE Common Commands

### Table 4-13 SGE common commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qconf -ae hostname</td>
<td>Adds an execution host.</td>
</tr>
<tr>
<td>qconf -de hostname</td>
<td>Deletes an execution host.</td>
</tr>
<tr>
<td>qconf -sel</td>
<td>Displays the list of execution hosts.</td>
</tr>
<tr>
<td>qconf -ah hostname</td>
<td>Adds an administration host.</td>
</tr>
<tr>
<td>qconf -dh hostname</td>
<td>Deletes an administration host.</td>
</tr>
<tr>
<td>qconf -sh</td>
<td>Displays the list of administration hosts.</td>
</tr>
<tr>
<td>qconf -as hostname</td>
<td>Adds a submission host.</td>
</tr>
<tr>
<td>qconf -ds hostname</td>
<td>Deletes a submission host.</td>
</tr>
<tr>
<td>qconf -ss</td>
<td>Displays the list of submission hosts.</td>
</tr>
<tr>
<td>qconf -ahgrp groupname</td>
<td>Adds a host user group.</td>
</tr>
<tr>
<td>qconf -mhgrp groupname</td>
<td>Modifies a host user group.</td>
</tr>
</tbody>
</table>
### Command

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>qconf -shgrp groupname</code></td>
<td>Display information about a host user group.</td>
</tr>
<tr>
<td><code>qconf -shgrpl</code></td>
<td>Display the list of host user groups.</td>
</tr>
<tr>
<td><code>qconf -aq queueName</code></td>
<td>Adds a cluster queue.</td>
</tr>
<tr>
<td><code>qconf -dqueueName</code></td>
<td>Deletes a cluster queue.</td>
</tr>
<tr>
<td><code>qconf -mq queueName</code></td>
<td>Modifies cluster queue configuration.</td>
</tr>
<tr>
<td><code>qconf -sq queueName</code></td>
<td>Displays cluster queue configuration.</td>
</tr>
<tr>
<td><code>qconf -sql</code></td>
<td>Displays the list of cluster queues.</td>
</tr>
<tr>
<td><code>qconf -ap PE_name</code></td>
<td>Adds a parallel environment (PE).</td>
</tr>
<tr>
<td><code>qconf -mp PE_name</code></td>
<td>Modifies a PE.</td>
</tr>
<tr>
<td><code>qconf -dp PE_name</code></td>
<td>Deletes a PE.</td>
</tr>
<tr>
<td><code>qconf -sp PE_name</code></td>
<td>Displays information about a PE.</td>
</tr>
<tr>
<td><code>qconf -spl</code></td>
<td>Displays the list of PE names.</td>
</tr>
<tr>
<td><code>qstat -f</code></td>
<td>Displays the execution host status.</td>
</tr>
<tr>
<td><code>qstat -u user</code></td>
<td>Queries user Jobs.</td>
</tr>
<tr>
<td><code>qhost</code></td>
<td>Displays information about the execution host resources.</td>
</tr>
</tbody>
</table>

### 4.3 Conda 4.3.16 Porting Guide (CentOS 7.6)

#### 4.3.1 Introduction

Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language.

Conda, as a package manager, helps you find and install packages. If you need a package that requires a different version of Python, you do not need to switch to a different environment manager, because Conda is also an environment manager. With just a few commands, you can set up a totally separate environment to run that different version of Python, while continuing to run your usual version of Python in your normal environment.

For more information about Conda, visit the [official Conda website](https://conda.io/).

Language: Python

Brief description: Open source package and environment management system.
Open-source protocol: 3-clause BSD License

**Recommended Version**

Conda 4.3.16

**4.3.2 Environment Requirements**

**Hardware Requirements**

Table 4-14 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

**Software Requirements**

Table 4-15 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>2.7.16</td>
<td><a href="https://www.python.org/downloads/release/python-2716/">https://www.python.org/downloads/release/python-2716/</a> Using the system Yum to install</td>
</tr>
<tr>
<td></td>
<td>2.7.5</td>
<td></td>
</tr>
</tbody>
</table>

**OS Requirements**

Table 4-16 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

**4.3.3 Paths for Software Porting**

This chapter lists the software installation paths involved in the Conda software porting.
### Table 4-17 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each installation package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

### 4.3.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install dependencies.</td>
<td>For details, see 4.3.4.1 Installing Dependencies.</td>
</tr>
<tr>
<td>3</td>
<td>Install Python.</td>
<td>For details, see Installing Python.</td>
</tr>
<tr>
<td>4</td>
<td>Install PIP.</td>
<td>For details, see Installing PIP.</td>
</tr>
</tbody>
</table>

#### 4.3.4.1 Installing Dependencies

#### Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to install the dependency:

```
yum install -y zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel python-devel
```

-----End
4.3.5 Obtaining the Source Code

Procedure

Step 1. The Conda software package is downloaded and installed through the link specified by the pip command. You do not need to download the source code package separately.

URL: https://pypi.tuna.tsinghua.edu.cn/simple

4.3.6 Compiling and Installing Conda

Step 1. Use PuTTY to log in to the server as the root user.

Step 2. Run the following commands to install Conda.

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple conda
```

Step 3. Run the following command to check whether Conda is installed successfully:

```
pip list
```

If conda is displayed in the command output, the installation is successful, as shown in Figure 4-3.

**Figure 4-3** Result example

```
Package    Version
----------  -------
certifi     2019.11.28
chardet     3.0.4
conda       4.3.16
Cython      0.29.15
enum34      1.1.9
future      0.18.2
h5py        2.10.0
idna        2.9
m2py        2.17
numpy       1.16.6
ont-tombo   1.5.1
pip         20.0.2
pycosat     0.6.3
requests    2.23.0
ruamel.ordereddict 0.4.14
ruamel.yaml   0.16.10
ruamel.yaml.clib 0.2.0
scipy       1.2.3
setuptools   44.8.0
six         1.14.0
tqdm        4.43.0
urllib3     1.25.8
wheel       0.34.2
```

4.4 MiniFE 2.2.0 Porting Guide (CentOS 7.6)
4.4.1 Introduction

MiniFE is a proxy application for unstructured implicit finite element code. It is similar to HPCCG and pHPCCG but provides a much more complete vertical covering of the steps in this class of applications.

For more information about miniFE, see MiniFE.

Programming language: C++

Brief description: a proxy application for unstructured implicit finite element code

Open-source license: LGPL-3.0

Recommended Version

MiniFE miniFE-2.2.0

4.4.2 Environment Requirements

Hardware Requirements

Table 4-18 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-19 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>MiniFE</td>
<td>miniFE-2.2.0</td>
<td><a href="https://github.com/Mantevo/miniFE/archive/2.2.0.tar.gz">https://github.com/Mantevo/miniFE/archive/2.2.0.tar.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-20 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Item</td>
<td>Version</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.4.3 Planning the Paths for Software Porting

**Table 4-21** lists the software installation paths involved in the MiniFE software porting.

**Table 4-21** Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/MINIFE</td>
<td>Installation path of MiniFE</td>
<td>The installation path provided here is only an example. A shared path is recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

### 4.4.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Procedure

**Table 4-22** Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Environment for the Cluster Scenario&quot; in <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
</tbody>
</table>
4.4.5 Obtaining the Source Code

Procedure

Step 1 Download the MiniFE installation package `miniFE-2.2.0.zip`.
URL: [https://github.com/Mantevo/miniFE/archive/2.2.0.tar.gz](https://github.com/Mantevo/miniFE/archive/2.2.0.tar.gz)

Step 2 Use SFTP to upload the MiniFE installation package to the `/path/to/MINI FE` directory on the server.

----End

4.4.6 Compiling and Installing MiniFE

Procedure

Step 1 Use PuTTY to log in to the server as the `root` user.

Step 2 Run the following commands to decompress the installation package:

```
tar -xvf miniFE-2.2.0.tar.gz
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```
cd /path/to/miniFE/miniFE-2.2.0/ref/src
```

Step 4 Run the following command to perform compilation:

```
make
```

----End

4.4.7 Running and Verifying MiniFE

Procedure

Step 1 Use PuTTY to log in to the server as the `root` user.

Step 2 Run the following command to run `miniFE.x` to generate a YAML file:

```
mpirun --allow-run-as-root -n 96 ./miniFE.x nx=300 verify_solution=1
```

**NOTE**

In the preceding command, `nx` indicates the X problem.

Check the values of **Total CG Mflops: 34377.9** and **Total Program Time: 29.567** in the generated YAML file.

- Iterations: 200
- Final Resid Norm: 0.00247174
- WAXPY Time: 1.36617
- WAXPY Flops: 4.8762e+10
- WAXPY Mflops: 35692.6
- DOT Time: 1.10191
- DOT Flops: 2.16e+10
- DOT Mflops: 19602.3
### 4.4.8 More Information

Official Mantecho website:

[http://www.mantevo.org/](http://www.mantevo.org/)

### 4.5 NPB 3.4 Porting Guide (CentOS 7.6)

#### 4.5.1 Introduction

NAS Parallel Benchmarks (NPB) are a set of benchmarks targeting performance evaluation of highly parallel supercomputers. They are developed and maintained by the NASA Advanced Supercomputing (NAS) Division (formerly the NASA Numerical Aerodynamic Simulation Program) based at the NASA Ames Research Center. The benchmarks are derived from computational fluid dynamics (CFD) applications and consist of five kernels and three pseudo-applications in the original "pencil-and-paper" specification (NPB 1). The benchmark suite has been extended to include new benchmarks for unstructured adaptive meshes, parallel I/O, multi-zone applications, and computational grids. Problem sizes in NPB are predefined and indicated as different classes. Reference implementations of NPB are available in commonly-used programming models like MPI and OpenMP (NPB 2 and NPB 3).

For more information about CAMx, visit the official NPB website.

Language: C/Fortran

Brief description: NPB is a set of benchmarks targeting performance evaluation of highly parallel supercomputers.

Open source protocol: NOSA

#### Recommended Versions

NPB 3.4

#### 4.5.2 Environment Requirements

##### Hardware Requirements

*Table 4-23* lists the hardware requirements.
Table 4-23 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-24 lists the software requirements.

Table 4-24 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPB</td>
<td>3.4</td>
<td><a href="https://www.nas.nasa.gov/publications/npb.html">https://www.nas.nasa.gov/publications/npb.html</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-25 lists the OS requirements.

Table 4-25 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.5.3 Planning the Paths for Software Porting

Table 4-26 lists the software installation paths involved in the NPB software porting.

Table 4-26 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>
### 4.5.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Procedure

**Table 4-27 Configuration process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Environment for the Cluster Scenario&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
</tbody>
</table>

### 4.5.5 Obtaining the Source Code

#### Procedure

**Step 1** Download the NPB installation package `NPB3.4.tar.gz`.

Download address: [https://www.nas.nasa.gov/publications/npb.html](https://www.nas.nasa.gov/publications/npb.html).

**Step 2** Use SFTP to upload the NPB installation package to the `/path/to/NPB` directory on the server.

----End

### 4.5.6 Compiling and Installing NPB

#### Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.
Step 2  Run the following command to create a main program installation directory:

```
mkdir -p /path/to/NPB
```

Step 3  Copy the installation package to the installation directory.

```
cp NPB3.4.tar.gz /path/to/NPB
```

Step 4  Run the following command to go to the main program installation directory:

```
cd /path/to/NPB
```

Step 5  Run the following command to decompress the installation package:

```
tar -xvf NPB3.4.tar.gz
```

Step 6  Run the following command to switch to the directory generated after the package is decompressed

```
cd NPB3.4
```

Step 7  Run the following command to generate a configuration file:

```
cd NPB3.4-MPI
cp config/make.def.template config/make.def
```

Step 8  Run the following command to load OPENMPI and GNU environmental variables:

```
export PATH=/path/to/OPENMPI/bin:/path/to/GNU/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENMPI/lib:/path/to/GNU/lib64:SLD_LIBRARY_PATH
```

Step 9  Run the following command to perform compilation.

```
make EP CLASS=D NPROCS=96
```

⚠️ NOTE

In the preceding command:

- **EP** indicates the benchmark. The NPB has 12 benchmarks: IS, EP, CG, MG, FT, BT, BT-10, SP, LU, UA, DC, and DT.
- **CLASS=D** indicates the problem size. NPB has eight problem sizes: S, W, A, B, C, D, E, and F.
- **NPROCS=96** indicates the number of cores.

You can select the benchmark, problem size, and number of cores based on the site requirements. For details about the parameters, see 4.5.8 More Information.

---End

4.5.7 Running and Verifying NPB

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Switch to the test directory.

```
   cd /path/to/NPB/NPB3.4/NPB3.4-MPI/bin
```
**Step 3** Run the following command to start the test. If the following information is displayed, the test is successful.

```bash
mpirun --allow-run-as-root --mca btl ^openib -np 96 ep.D.x
```

**Figure 4-4** Result example

![Result example image](image)

---End

### 4.5.8 More Information

- 12 NPB benchmarks:

**Table 4-28 NPB benchmarks**

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Full Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS</td>
<td>Integer Sort</td>
<td>It is used to solve two-dimensional large integer sorting based on the bucket sort, and requires full-switching communication.</td>
</tr>
<tr>
<td>Benchmark</td>
<td>Full Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EP</td>
<td>Embarrassingly Parallel</td>
<td>It is used to calculate the Gauss pseudo random number and mainly performs floating-point number calculation. The prominent feature of EP is that no communication between processors is performed. Therefore, different results obtained through interconnection show the same execution characteristics.</td>
</tr>
<tr>
<td>CG</td>
<td>Conjugate Gradient</td>
<td>It is used to solve the approximation of the minimum eigenvalue of a large sparse symmetric positive definite matrix.</td>
</tr>
<tr>
<td>MG</td>
<td>MultiGrid</td>
<td>It is used to monitor short- and long-distance communication. MG is a simplified multi-grid core.</td>
</tr>
<tr>
<td>FT</td>
<td>Fast Fourier Transform</td>
<td>It is used to detect long-distance communication and solve 3-dimensional partial differential equation using fast Fourier transform.</td>
</tr>
<tr>
<td>BT</td>
<td>Block Tri-Diagonal</td>
<td>It is used to solve the tri-diagonal equation set.</td>
</tr>
<tr>
<td>BT-IO</td>
<td>Block Tri-Diagonal-IO</td>
<td>It is used to test different parallel I/O technologies.</td>
</tr>
<tr>
<td>SP</td>
<td>Scalar Penta-diagonal</td>
<td>It is used to solve the penta-diagonal equation set.</td>
</tr>
</tbody>
</table>
### Benchmark

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Full Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LU</td>
<td>Lower Upper Triangular</td>
<td>It is used to solve sparse block equations based on symmetric super-relaxation method.</td>
</tr>
<tr>
<td>UA</td>
<td>Unstructured Adaptive</td>
<td>It is used to test unstructured adaptive networks, dynamic and irregular memory access.</td>
</tr>
<tr>
<td>DC</td>
<td>Data Cube</td>
<td>It is used to test data movement.</td>
</tr>
<tr>
<td>DT</td>
<td>Data Traffic</td>
<td></td>
</tr>
</tbody>
</table>

- Eight NPB problem sizes:

**Table 4-29 NPB problem sizes**

<table>
<thead>
<tr>
<th>Problem Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>The S (sample) class is for sample programs.</td>
</tr>
<tr>
<td>W</td>
<td>The W (workstation) class is usually used for workstations.</td>
</tr>
<tr>
<td>A-F</td>
<td>The size of A class is the smallest and increases in ascending order.</td>
</tr>
</tbody>
</table>

### 4.6 MiniDFT 1.1.1 Porting Guide (CentOS 7.6)

#### 4.6.1 Introduction

MiniDFT is a plane-wave density functional theory (DFT) mini-app for modeling materials. It is extracted from Quantum Espresso (QE).

For more information about MiniDFT, visit the [official MiniDFT website](#).

Language: Fortran

Brief description: a mini-app for modeling materials.

Open-source license: GPL

**Recommended Version**

MiniDFT MiniDFT-1.1.1
4.6.2 Environment Requirements

Hardware Requirements

Table 4-30 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-31 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>MiniDFT</td>
<td>1.1.1</td>
<td><a href="https://github.com/NERSC/MiniDFT/releases">https://github.com/NERSC/MiniDFT/releases</a></td>
</tr>
<tr>
<td>OpenBLAS</td>
<td>0.3.6</td>
<td><a href="https://github/xianyi/OpenBLAS/tree/v0.3.6">https://github/xianyi/OpenBLAS/tree/v0.3.6</a></td>
</tr>
<tr>
<td>fftw</td>
<td>3.3.8</td>
<td><a href="http://www.fftw.org/download.html">http://www.fftw.org/download.html</a></td>
</tr>
<tr>
<td>scalapack</td>
<td>2.1.0</td>
<td><a href="http://www.netlib.org/scalapack/">http://www.netlib.org/scalapack/</a></td>
</tr>
<tr>
<td>Test case</td>
<td>small.in</td>
<td>Test case provided by the software</td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-32 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.6.3 Planning the Paths for Software Porting

Table 4-33 lists the software installation paths involved in the MiniDFT software porting.
### Table 4-33 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see <strong>Planning the Installation Paths</strong> in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/FFTW</td>
<td>Installation path of fftw</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/SCALAPACK</td>
<td>Installation path of scalapack</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/MINIDFT</td>
<td>Installation path of MiniDFT</td>
<td></td>
</tr>
</tbody>
</table>

### 4.6.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Procedure

### Table 4-34 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Installing OpenBLAS</td>
<td>For details, see 4.6.4.1 Installing OpenBLAS.</td>
</tr>
<tr>
<td>3</td>
<td>Installing fftw</td>
<td>For details, see 4.6.4.2 Installing fftw.</td>
</tr>
<tr>
<td>No.</td>
<td>Operation</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4</td>
<td>Installing scalapack</td>
<td>For details, see <a href="#">4.6.4.3 Installing scalapack</a>.</td>
</tr>
</tbody>
</table>

### 4.6.4.1 Installing OpenBLAS

**Procedure**

**Step 1**  Use PuTTY to log in to the server as the **root** user.

**Step 2**  Run the following command to install BLAS LAPACK:

```
yum install blas64.aarch64 blas64-devel.aarch64 lapack64.aarch64 lapack64-devel.aarch64 -y
```

**NOTE**

If you use a non-default compiler to install the OpenBLAS, download the OpenBLAS and use the specified compiler to install it. Specify the OpenBLAS path for compilation and installation in [Compiling and Installing](#).

**Step 3**  Run the following command to decompress the OpenBLAS installation package:

```
tar -xvf OpenBLAS-0.3.6.tar.gz
```

**Step 4**  Run the following command to switch to the directory generated after the package is decompressed:

```
cd OpenBLAS-0.3.6
```

**Step 5**  Run the following commands to perform the configuration:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

**Step 6**  Run the following commands to perform compilation and installation:

```
make
make PREFIX=/path/to/OPENBLAS/install
```

---End

### 4.6.4.2 Installing fftw

**Procedure**

**Step 1**  Use PuTTY to log in to the server as the **root** user.

**Step 2**  Run the following command to decompress the fftw installation package:

```
tar -zxvf fftw-3.3.8.tar.gz
```
Step 3  Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd fftw-3.3.8
```

Step 4  Run the following command to perform configuration:

```bash
./configure --prefix=/path/to/FFTW
CC=gcc FC=gfortran F77=gfortran --enable-openmp --enable-threads --enable-threads --enable-shared=yes
```

Step 5  Run the following commands to perform compilation and installation:

```bash
make -j
make install
```

----End

4.6.4.3 Installing scalapack

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to decompress the scalapack installation package:

```bash
tar -xvf scalapack-2.1.0.tgz
```

Step 3  Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd scalapack-2.1.0
```

Step 4  Run the following command to copy and paste the file:

```bash
cp SLmake.inc.example SLmake.inc
```

Step 5  Modify the SLmake.inc file.

1. ```bash
   vim SLmake.inc
   ```
2. Press `I` to enter the editing mode and create the computing instance files.

   ```bash
   BLASLIB = -L/path/to/OPENBLAS/lib -lopenblas
   LAPACKLIB = -L/path/to/OPENBLAS/lib -lopenblas
   ```
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

Step 6  Run the following commands to perform compilation:

```bash
make
```

Step 7  Run the following command to copy the generated link library to the installation directory:

```bash
cp libscalapack.a /path/to/SCALAPACK
```

Step 8  Run the following commands to add the environment variables:

```bash
export LD_LIBRARY_PATH=/path/to/SCALAPACK
```

----End
4.6.5 Obtaining the Source Code

Procedure

Step 1  Download the MiniDFT installation package MiniDFT-MiniDFT-1.1.1.tar.gz.
        Download address: https://github.com/NERSC/MiniDFT/releases

Step 2  Use SFTP to upload the MiniDFT installation package to the /path/to/MINIDFT
directory on the server.

----End

4.6.6 Compiling and Installing MiniDFT

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to decompress the MiniDFT installation package:
        tar -xzvf MiniDFT-MiniDFT-1.1.1.tar.gz

Step 3  Run the following command to switch to the directory generated after the
        package is decompressed:
        cd MiniDFT-MiniDFT-1.1.1/src

Step 4  Run the following commands to modify the Makefile file:
        1.  vi Makefile
        2.  Press I to enter the insert mode and modify information in bold in the
            Makefile file.
           FFTW_INCL = -I/path/to/FFTW/include
           FFTW_LIBS = -L/path/to/SCALAPACK/libscalapack.a -Wl,--start-group /path/to/FFTW/lib/libfftw3.a /path/to/FFTW/lib/libfftw3_threads.a /path/to/OPENBLAS/lib/libopenblas.a -Wl,--end-group
           OPENBLAS_INC = -I/path/to/OPENBLAS/include
           OPENBLAS_LIB = /path/to/OPENBLAS/lib
           SCALAPACK_LIBS = /path/to/SCALAPACK/libscalapack.a
           CC = mpicc
           CFLAGS = -D__FFTW3 -D__para11e1 -D__SCALAPACK
           FC = mpifort
           FFLAGS = -cpp -no-pie
           LD = mpifort
           DFLAGS += -D__OPENMP
           FFLAGS += -fopenmp
           LDFLAGS += -fopenmp
           include Makefile.base
        3.  Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 5  Run the following commands to perform compilation and installation:
4.6.7 Running and Verifying MiniDFT

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to perform parallel computing:

1. Run the following command to go to the example folder:
   ```
cd /path/to/MINIDFT/MiniDFT-MiniDFT-1.1.1/benchmark
   ```
2. Run the following command to configure the threads:
   ```
   export OMP_NUM_THREADS=1
   ```
3. Run the following commands as user root to start the test:
   ```
   mpirun -np 96 --allow-run-as-root -mca btl ^openib ../src/mini_dft -in small.in -nbgrp 4 > small.log
   ```

In the `small.log` file, check the value of `Benchmark_Time` (in seconds). A smaller value indicates higher performance. **Figure 4-5** shows the command output.

![Figure 4-5 Result example](image)

4.6.8 Troubleshooting

An Error Is Reported During MiniDFT Compilation

**Symptom**

An error message "pwscf.f90:(.text+0x20): undefined reference to `dfftw_init_threads" is displayed during MiniDFT compilation.

**Possible causes**

fftw does not have the threads module.
Procedure
Recompile fftw and add the --enable-threads parameter during compilation. For details, see Step 4.

Problem 2: An Error is Reported When the MiniDFT Is Run

Symptom
When large.in is run, "mpirun noticed that process rank 84 with PID 0 on node XA320V2-48 exited on signal 11 (Segmentation fault)" is reported.

Possible causes
The software is memory consuming, causing shortage of memory resources.

Procedure
Increase the memory capacity or use a small small.in computing instance.

4.6.9 More Information

MiniDFT official installation guide:
https://github.com/NERSC/MiniDFT

4.7 miniGhost v1.0 Porting Guide (CentOS 7.6)

4.7.1 Introduction

miniGhost represents 3D nearest neighbor halo-exchange communication that is present in a lot of HPC code. A broad range of scientific computation involves the use of difference stencils. In a parallel computing environment, this computation is typically implemented by decomposing the spacial domain, inducing a "halo exchange" of process-owned boundary data. This approach adheres to the Bulk Synchronous Parallel (BSP) model. Because commonly available architectures provide strong inter-node bandwidth relative to latency costs, many codes 'bulk up' these messages by aggregating data into a message as a means of reducing the number of messages.

For more information on miniGhost, visit the official miniGhost website.

Programming language: Fortran

Brief description: miniGhost represents 3D nearest neighbor halo-exchange communication that is present in a lot of HPC code.

Open-source license: LGPL-3.0

Recommended Version

miniGhost v1.0
4.7.2 Environment Requirements

Hardware Requirements

Table 4-35 lists the hardware requirements.

Table 4-35 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-36 lists the software requirements.

Table 4-36 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>miniGhost</td>
<td>miniGhost v1.0</td>
<td><a href="https://github.com/Mantevo/miniGhost">https://github.com/Mantevo/miniGhost</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-37 lists the OS requirements.

Table 4-37 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.7.3 Planning the Paths for Software Porting

Table 4-38 lists the software installation paths involved in the miniGhost software porting.
### 4.7.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Configuration Process

**Table 4-39 Configuration process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Environment for the Cluster Scenario&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
</tbody>
</table>

### 4.7.5 Obtaining the Source Code

#### Procedure

**Step 1** Download the miniGhost installation package `miniGhost-master.zip`.

**URL:** [https://github.com/Mantevo/miniGhost](https://github.com/Mantevo/miniGhost)
Step 2 Use the SFTP tool to upload the miniGhost installation package to the
`/path/to/MINIGHOST` directory on the server.

----End

4.7.6 Compiling and Installing miniGhost

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the miniGhost installation package:

```bash
unzip miniGhost-master.zip
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd /path/to/MINIGHOST/miniGhost-master/ref
```

Step 4 Run the following command to create the Makefile file:

```bash
mv Makefile Makefile.bak
```

```bash
cp makefile.mpi.gnu Makefile
```

Step 5 Run the following command to modify the Makefile file:

1. `vi Makefile`

2. Press i to enter the insert mode and modify the file as follows:

   ```bash
   PROTOCOL = -D_MG_MPI
   MPI_LOC = /path/TO/OPENMPI
   MPI_INCLUDE = -I$(MPI_LOC)/include
   # State precision: -D_REAL4 and/or -D_REAL8.
   REAL_PRECISION = -D_MG_REAL8
   # Integer precision: -D_INT4 and/or -D_INT8.
   INT_PRECISION = -D_MG_INT4
   # Compilers
   FC=$(MPI_LOC)/bin/mpif90
   CC=$(MPI_LOC)/bin/mpicc
   CFLAGS = $(PROTOCOL) $(REAL_PRECISION) $(INT_PRECISION)
   # C main calling Fortran subroutine:
   FFLAGS = $(PROTOCOL) $(REAL_PRECISION) $(INT_PRECISION)
   # Optimization
   OPT_F = -O3
   FFLAGS += $(OPT_F)
   #FFLAGS += $(MPI_INCLUDE)
   # Free-form Fortran source code:
   FFLAGS += -ffree-form -ffree-line-length-none
   # Array bounds checking: (expensive!)
   #FFLAGS += -fbounds-check
   # Compile to include checkpointing capability.
   FFLAGS += -D_MG_CHECKPT
   NCPATH=
   LNCDF=-L$(NCPATH)/lib
   LD=$(CC)
   LDFLAGS=$(CC) $(FFLAGS) -L/path/to/GNU/gcc9.1/lib64/ -lgfortran -L/path/to/OPENMPI/openmpi-4.0.1-gcc9.1/lib -lmpi_mpifh
   #LDFLAGS=$(CC) $(FFLAGS)
   #LDFLAGS=$(CC) $(FFLAGS)
   LIBS=
   include make_targets
   ```
3. Press Esc, type :wq!, and press Enter to save the file and exit.

Step 6 Run the following command to perform compilation:

```bash
make
```

**NOTE**

Parallel compilation (by using the `make -j` command) is not supported. Run the `make` command to perform compilation.

----End

### 4.7.7 Running and Verifying miniGhost

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to start miniGhost.x:

```bash
mpirun -mca btl ^openib -x LD_LIBRARY_PATH --allow-run-as-root -np 96 ./
miniGhost.x
```

Check the value of each parameter in the generated `results.yaml` file.

**Figure 4-6** shows the command output.

**Figure 4-6 Result example**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFLOPS_Total</td>
<td>1.991E+01</td>
</tr>
<tr>
<td>GFLOPS_per_process</td>
<td>1.556E-01</td>
</tr>
<tr>
<td>GFLOPS_stencil_Total</td>
<td>1.991E+01</td>
</tr>
<tr>
<td>GFLOPS_stencil_per_process</td>
<td>1.556E-01</td>
</tr>
<tr>
<td>FLOPS_Total</td>
<td>3.200E+11</td>
</tr>
<tr>
<td>FLOPS_Sums</td>
<td>2.560E+11</td>
</tr>
<tr>
<td>FLOPS_Divide</td>
<td>6.400E+10</td>
</tr>
<tr>
<td>Number_spikes</td>
<td>1</td>
</tr>
</tbody>
</table>

----End

### 4.7.8 More Information

Official Mantevo website:

http://www.mantevo.org/

### 4.8 BLAST 2.9.0 Porting Guide (CentOS 7.6)

#### 4.8.1 Introduction

Basic Local Alignment Search Tool (BLAST) is an analysis tool used for similarity comparison in the protein database or DNA database. The BLAST program can quickly find the similarity between the object sequence and that recorded in public databases. The score in the BLAST result is a statistical description of similarity.
BLAST uses a local algorithm to obtain sequences that are similar in two sequences. For details about the BLAST algorithm, see the BLAST course of NCBI.

BLAST compares one or more sequences (in any form) to one or more nucleic acid or protein sequence libraries. BLAST can also find sequences with matched gaps.

Based on the method published by Altschul et al. on J. Mol. Biol (J. Mol. Biol. 215: 403-410 (1990)), BLAST checks/analyzes the homology between queried sequences and those recorded in sequence databases. Compared with the initial BLAST, BLAST 2.0 provided by NCBI adopts gap penalty in sequence alignment. BLAST can process any number of sequences, including protein sequences and nucleic acid sequences. Multiple databases of the same type can be selected, that is, either all protein databases or all nucleic acid databases. The queried sequence and the invoked database may be a combination of any form. You can analyze a nucleic acid sequence or protein sequence with a protein or nucleic acid database.

Software packages such as GCG and EMBOSS contain five types of BLAST:

- **BLASTP** is used to query protein sequences in protein databases. It compares each queried sequence with each sequence recorded in databases one by one.
- **BLASTX** is used to query nucleic acid sequences in protein databases. First, it translates the nucleic acid sequence into a protein sequence (translates one nucleic acid sequence into six possible proteins), and then performs a one-to-one protein sequence comparison on each of the protein sequences.
- **BLASTN** is used to query nucleic acid sequences in nucleic acid databases. It compares each known sequence in databases with each queried nucleic acid sequence one by one.
- **TBLASTN** is used to query protein sequences in nucleic acid databases. In contrast to BLASTX, it translates nucleic acid sequences in databases into protein sequences, and then compares them with the queried sequences.
- **TBLASTX** is used to query nucleic acid sequences in nucleic acid databases. It translates both the nucleic acid sequences in databases and the queried nucleic acid sequences into proteins (each nucleic acid sequence produces six possible protein sequences), which produces 36 comparison arrays each time.

For more information about BLAST, visit the [official BLAST website](http://www.ncbi.nlm.nih.gov/BLAST).

Programming language: C++

Brief description: An analysis tool used for similarity comparison in the protein database or DNA database.

Open-source license: PUBLIC DOMAIN

**Recommended Version**

BLAST 2.9.0

**4.8.2 Environment Requirements**

**Hardware Requirements**

*Table 4-40* lists the hardware requirements.
Table 4-40 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-41 lists the software requirements.

Table 4-41 Software Requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAST</td>
<td>2.9.0</td>
<td><a href="https://ftp.ncbi.nlm.nih.gov/blast/executeables/blast+/2.9.0/">https://ftp.ncbi.nlm.nih.gov/blast/executeables/blast+/2.9.0/</a></td>
</tr>
<tr>
<td>LMDB</td>
<td>0.9.24</td>
<td><a href="https://github.com/LMDB/lmdb/releases">https://github.com/LMDB/lmdb/releases</a></td>
</tr>
<tr>
<td>Test case 1</td>
<td>protein_query.fastas</td>
<td><a href="https://gitlab.com/arm-hpc/packages/uploads/1acce57b88cc76f6c89a087e449456b5/protein_query.fasta">https://gitlab.com/arm-hpc/packages/uploads/1acce57b88cc76f6c89a087e449456b5/protein_query.fasta</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-42 lists the OS requirements.

Table 4-42 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
</tbody>
</table>

4.8.3 Paths for Software Porting

This section lists the software installation paths involved in the BLAST software porting.
Table 4-43 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/LMDB</td>
<td>Installation path of LMDB</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/BLAST</td>
<td>Installation path of BLAST</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/CASE</td>
<td>Path for storing the test cases</td>
<td></td>
</tr>
</tbody>
</table>

4.8.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Procedure

Table 4-44 Procedure

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing LMDB</td>
<td>Install tools by following the instructions in section Step 5.</td>
</tr>
</tbody>
</table>

4.8.5 Obtaining the Source Code

Procedure

Step 1  Download the BLAST installation package ncbi-blast-2.9.0+-src.tar.gz.

Step 2 Use SFTP to upload the BLAST installation package to the path/to/BLAST directory on the server.

----End

4.8.6 Compiling and Installing BLAST

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress and configure the BLAST path:

```bash
tar xzf ncbi-blast-2.9.0+-src.tar.gz
cd ncbi-blast-2.7.1+-src
export BLAST_TOP=`pwd`
```

Step 3 Run the following command to configure the compiler:

```bash
export CC=gcc
export CXX=g++
```

Step 4 Use an SFTP tool to upload lmdb-LMDB_0.9.s24.tar.gz to the /path/to/BLAST/ncbi-blast-2.9.0+-src directory.

Step 5 Run the following command to decompress and configure the LMDB path:

```bash
cd $BLAST_TOP
tar -xvf lmdb-LMDB_0.9.s24.tar.gz
cd lmdb-LMDB_0.9.24
export LMDB_PATH=$BLAST_TOP/lmdb-LMDB_0.9.24
```

Step 6 Run the following commands to compile and install BLAST:

```bash
cd $BLAST_TOP/c++
./configure
cd ReleaseMT/build
make all_r -j40
```

Step 7 After the installation is successful, the executable file is generated in the $BLAST_TOP/c++/ReleaseMT/bin directory. You can run the following command to view the file:

```bash
cd $BLAST_TOP/c++/ReleaseMT/bin
ls blastp
blastp
```

----End
4.8.7 Running and Verifying BLAST

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use the SFTP to upload the test case to the /path/to/CASE directory.

Step 3 Run the following commands to decompress and rename Swissprot Proteins data:

```
cd /path/to/CASE
gzip -d swissprot.gz
mv swissprot swissprot.fa
```

Step 4 Run the following command to save all files to the data directory:

```
mkdir data
mv swissprot.fa data/
mv protein_query.fasta data/
```

Step 5 Run the following command to compile the test case data.

```
export PATH=$BLAST_TOP/c++/ReleaseMT/bin:$PATH
makeblastdb -in data/swissprot.fa -dbtype prot
```

After the command is executed, the following four files are generated in the data directory. You can run the `ls` command to view the files.

1. swissprot.fa
2. swissprot.fa.phr
3. swissprot.fa.pin
4. swissprot.fa.psq

Step 6 Run the following command to run the test:

```
{ time blastp -query data/protein_query.fasta -db data/swissprot.fa ; } 2>&1 | tee output_protein_alignments.txt
```

View the real value in the last time command output in the log. The value indicates $x$ minutes $y$ seconds. The lower the value is, the better the performance is.

**Figure 4-7** shows the command output.

**Figure 4-7** Test example

```
real 0m2.698s
user 0m2.686s
sys 0m0.010s
```

----End
4.8.8 More Information

Official Arm website:


4.9 BWA 0.7.17 Porting Guide (CentOS 7.6)

4.9.1 Introduction

Burrows-Wheeler Aligner (BWA) is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW, and BWA-MEM. The first algorithm is designed for Illumina sequence reads up to 100 bp. The rest two are for longer sequences ranged from 70 bp to 1Mbp. BWA-MEM and BWA-SW share similar features, such as long-read support and split alignment. However, BWA-MEM is generally recommended for high-quality queries as it is faster and more accurate. Compared with BWA-backtrack, BWA-MEM has better performance in 70–100 bp Illumina reads.

For more information, visit the official BWA website.

Programming language: C

Brief description: gene comparison

Open-source license: GPL 3.0

Recommended Version

BWA 0.7.17

4.9.2 Environment Requirements

Hardware Requirements

Table 4-45 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-46 lists the software requirements.
### Table 4-46 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>BWA</td>
<td>0.7.17</td>
<td><a href="https://sourceforge.net/projects/bio-bwa/files/">https://sourceforge.net/projects/bio-bwa/files/</a></td>
</tr>
<tr>
<td>Compute instance file 1</td>
<td>B17NC_R1.fq.gz</td>
<td><a href="https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz">https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz</a></td>
</tr>
<tr>
<td>Compute instance file 2</td>
<td>B17NC_R2.fq.gz</td>
<td><a href="https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R2.fastq.gz">https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R2.fastq.gz</a></td>
</tr>
</tbody>
</table>

### OS Requirements

Table 4-47 lists the OS requirements.

### Table 4-47 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.9.3 Planning the Paths for Software Porting

Table 4-48 lists the software installation paths involved in the BWA software porting.

### Table 4-48 Paths for software porting

<table>
<thead>
<tr>
<th>N o.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see <a href="https://www.huawei.com/en">Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide</a></td>
</tr>
<tr>
<td>2</td>
<td>/path/to/BWA</td>
<td>Installation path of BWA</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths</td>
</tr>
</tbody>
</table>
No. | Software Installation Path | Description | Remarks |
--- | --- | --- | --- |
3 | /path/to/CASE | Path for storing the BWA compute instances to be tested | paths used in the commands in this document are examples only. Use the actual paths planned during the installation process. |

### 4.9.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.

**Procedure**

<table>
<thead>
<tr>
<th>Table 4-49 Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

### 4.9.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the BWA installation package bwa-0.7.17.tar.bz2.

URL: [https://sourceforge.net/projects/bio-bwa/files/](https://sourceforge.net/projects/bio-bwa/files/)

**Step 2** Use SFTP to upload the installation package to the /path/to/BWA directory on the server.

----End

### 4.9.6 Compiling and Installing BWA

**Prerequisites**

You have obtained SSE2NEON.h from [https://github.com/jratcliff63367/sse2neon](https://github.com/jratcliff63367/sse2neon).
Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to switch to the directory, in which the BWA installation package is stored:

```
cd /path/to/BWA
```

**Step 3** Decompress the BWA installation package.

```
tar -xvf bwa-0.7.17.tar.bz2
```

**Step 4** Switch to the folder generated after the BWA installation package is decompressed.

```
cd bwa-0.7.17
```

**Step 5** Copy `sse2neon.h` to the installation directory of the compiler.

```
cp SSE2NEON.h /path/to/GNU/lib/gcc/aarch64-unknown-linux-gnu/9.1.0/include
```

**Step 6** Modify the BWA **Makefile** file.

1. `vi Makefile`
2. Press I to enter the insert mode and modify line 14 in the **Makefile** file. Pay attention to the information in bold.
   ```
   INCLUDES= -I/path/to/GNU/lib/gcc/aarch64-unknown-linux-gnu/9.1.0/include
   ```

   **NOTE**
   Change the `include` directory to the `include` directory of the GNU compiler.
3. Press Esc, enter `:wq!`, and press Enter to save the file and exit.

**Step 7** Modify the **ksw.c** file.

1. `vi ksw.c`
2. Press I to enter the insert mode and modify line 29 in the **ksw.c** file. Pay attention to the information in bold.
   ```
   include <SSE2NEON.h>
   ```
3. Press Esc, enter `:wq!`, and press Enter to save the file and exit.

**Step 8** Run the following command to complete compilation:

```
maker
```

**Step 9** Load the BWA environment.

```
export PATH=/path/to/BWA/bwa-0.7.17:$PATH
```

---End

### 4.9.7 Running and Verifying BWA

Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.
Step 2 Create a BWA working directory.
```
mkdir -p /path/to/CASE
```

Step 3 Upload the compute instance files in Table 4-46 to the working directory.

Step 4 Switch to the directory, in which the compute instance files are stored.
```
cd /path/to/CASE
```

Step 5 Decompress the compute instance files:
```
gzip -d B17NC_R1.fq.gz
gzip -d B17NC_R2.fq.gz
```

Step 6 Create an index file.
```
bwa index -a bwtsw hg19.fa
```

Step 7 Run the compute instances.
```
bwa mem -t 96 hg19.fa B17NC_R1.fq B17NC_R2.fq > bwa.sam 2> bwa_log
```

Check the value of Real time (in seconds) in the `bwa_log` log. A smaller value indicates better performance.

Figure 4-8 shows the command output.

Figure 4-8 Test example
```
[M:mem_process_seq] Processed 3671839 reads in 830.539 CPU sec, 10.435 real sec
[main] Version: 0.7.17-r1188
[main] Real Time: 56.289 sec; CPU: 1073.919 sec
```

4.10 CNVnator 0.4.1 Porting Guide (CentOS 7.6)

4.10.1 Introduction

CNVnator is a tool used for copy number variation (CNV) discovery and genotyping from read-depth (RD) analysis of personal genome sequencing in the field of gene sequencing. CNVnator is based on combining the established mean-shift approach with additional refinements (multiple-bandwidth partitioning and GC correction) to broaden the range of discovered CNVs. Furthermore, CNVnator is complementary in a straightforward way to split-read and read-pair approaches: It misses CNVs created by retrotransposable elements, but more than half of the validated CNVs that it identifies are not detected by split-read or read-pair.

For more information about CNVnator, visit the official CNVnator website.

Programming language: C++

Brief description: a tool used for copy number variation (CNV) discovery and genotyping from read-depth (RD) analysis of personal genome sequencing

Open source protocol: MIT
Recommended Version

CNVnator V0.4.1

4.10.2 Environment Requirements

Hardware Requirements

Table 4-50 lists the hardware requirements.

Table 4-50 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-51 lists the software requirements.

Table 4-51 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNVnator</td>
<td>V0.4.1</td>
<td><a href="https://github.com/abyzovlab/CNVnator">https://github.com/abyzovlab/CNVnator</a></td>
</tr>
<tr>
<td>CERN ROOT</td>
<td>V6.18</td>
<td><a href="https://root.cern.ch/node/2998">https://root.cern.ch/node/2998</a></td>
</tr>
<tr>
<td>CMake</td>
<td>3.9.2</td>
<td><a href="https://cmake.org/files/v3.3/cmake-3.3.2.tar.gz">https://cmake.org/files/v3.3/cmake-3.3.2.tar.gz</a></td>
</tr>
<tr>
<td>SAMtools</td>
<td>1.10</td>
<td><a href="http://www.htslib.org/download/">http://www.htslib.org/download/</a></td>
</tr>
<tr>
<td>Test case</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>file</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-52 lists the OS requirements.

Table 4-52 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
4.10.3 Planning the Paths for Software Porting

Table Paths for software porting lists the software installation paths involved in the CNVnator software porting.

**Table 4-53 Paths for software porting**

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/SAMTOOLS</td>
<td>Installation path of SAMtools</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/SOURCE</td>
<td>Installation path of ROOT tool package</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/CNVNATOR</td>
<td>Installation path of CNVnator</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/CASE</td>
<td>Path for storing the CNVnator test cases</td>
<td></td>
</tr>
</tbody>
</table>

4.10.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.

**Configuration Process**

**Table 4-54 Configuration process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see “Setting Up the Single-Node System Environment” in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install the pre-installation packages and dependencies.</td>
<td>For details, see 4.10.4.1 Installing the Pre-installation Packages and Dependencies.</td>
</tr>
<tr>
<td>No.</td>
<td>Operation</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>Install CMake.</td>
<td>For details, see 4.10.4.2 Installing CMake.</td>
</tr>
<tr>
<td>4</td>
<td>Install SAMtools.</td>
<td>For details, see 4.10.4.3 Installing SAMtools.</td>
</tr>
<tr>
<td>5</td>
<td>Install the ROOT tool package.</td>
<td>For details, see 4.10.4.4 Installing the ROOT Tool Package.</td>
</tr>
</tbody>
</table>

4.10.4.1 Installing the Pre-installation Packages and Dependencies

Procedure

**Step 1** Use PuTTY to log in to the server as the `root` user.

**Step 2** Run the following commands to install the pre-installation packages and dependencies:
```
yum install -y libcurl-devel.aarch64 cmake binutils \
libX11-devel libXpm-devel libXft-devel libXext-devel \
openssl-devel pcre-devel mesa-libGL-devel \
mesa-libGLU-devel glew-devel ftgl-devel mysql-devel \
fftw-devel cfitsio-devel graphviz-devel \
avahi-compat-libdns_sd-devel libldap-dev python-devel \
libxml2-devel gsl-static
```

----End

4.10.4.2 Installing CMake

Procedure

**Step 1** Use PuTTY to log in to the server as the `root` user.

**Step 2** Run the following command to decompress the CMake installation package:
```
tar -xvf cmake-3.15.2.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after the package is decompressed:
```
cd cmake-3.15.2
```

**Step 4** Run the following command to perform configuration:
```
./configure --prefix=/path/to/CMAKE
```

**Step 5** Run the following commands to perform compilation and installation:
make
make install

**Step 6** Run the following command to set the CMake environment variables:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

### 4.10.4.3 Installing SAMtools

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to decompress the SAMtools installation package:

```
tar -xvf samtools-1.10.tar.bz2
```

**Step 3** Run the following command to switch to the directory generated after the package is decompressed:

```
cd samtools-1.10
```

**Step 4** Run the following command to perform installation:

```
yum install -y bzip2-devel bzip2-libs bzip2
```

**Step 5** Run the following command to perform configuration:

```
./configure --prefix=/path/to/SAMTOOLS
```

**Step 6** Run the following commands to perform compilation and installation:

```
make all all-htslib
make install install-htslib
```

**Step 7** Run the following commands to set the SAMtools environment variables:

```
export PATH=/path/to/SAMTOOLS/bin:$PATH
export LD_LIBRARY_PATH=/path/to/SAMTOOLS/lib:$LD_LIBRARY_PATH
```

----End

### 4.10.4.4 Installing the ROOT Tool Package

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Upload the ROOT tool package to the `/path/to/SOURCE` directory.

**Step 3** Run the following command to decompress the installation package `root_v6.18.00.source.tar.gz`:

```
cd /path/to/SOURCE
tar -xvf root_v6.18.00.source.tar.gz
```
Step 4  Run the following commands to create the root_build directory in the /path/to/SOURCE directory. ROOT cannot be compiled in the source code.

```bash
mkdir root_build
cd root_build
cmake ../root-6.18.00
```

Step 5  If the environment cannot be connected to the external network, you need to manually download the required installation packages to the corresponding directory. The following installation packages are contained in the ROOT tool package.

- Save the `davix-embedded-0.6.7.tar.gz` installation package to the `root_build/builtins/davix/DAVIX-prefix/src` directory.
- Save the `vdt-0.4.2.tar.gz` installation package to the `root_build/VDT-prefix/src` directory.
- Save the `tbb2019_U1.tar.gz` installation package to the `root_build/TBBefix/src` directory.

Step 6  Run the following command to modify the CMakeCache.txt file in `<root_build>`. Open the CMakeCache.txt file in `<root_build>` and set whether to install the Clad to OFF.

1. `vi CMakeCache.txt`
2. Press `I` to enter the editing mode and set whether to install clad to OFF in the CMakeCache.txt file.

```bash
clad BOLL=OFF
```
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

Step 7  Run the following command to perform compilation:

```bash
make -j32
```

Step 8  Run the following commands to load environment variables:

```bash
export PATH=/path/to/SOURCE/root_build/bin:$PATH
export LD_LIBRARY_PATH=/path/to/SOURCE/root_build/lib:SLD_LIBRARY_PATH
```

4.10.5 Obtaining the Source Code

Procedure

Step 1  Download the CNVnator installation package `CNVnator_v0.4.1.zip`.

URL: https://github.com/abyzovlab/CNVnator

Step 2  Use SFTP to upload the CNVnator installation package to the /path/to/CNVNATOR directory on the server.

----End
4.10.6 Compiling and Installing CNVnator

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to go to the CNVnator installation directory and create the dependency link:

```
cd /path/to/CNVNATOR
unzip CNVnator_v0.4.1.zip
cd CNVnator_v0.4.1/src
ln -s /path/to/SAMTOOLS/samtools-1.10 samtools
ln -s /path/to/SOURCE/root_build root_build
```

Step 3  Run the following command to modify the Makefile file:

```
1. vim Makefile
2. Press I to enter the editing mode and add the following content to the beginning of line 4 in the Makefile file:
   ROOTSYS=root_build
3. Press Esc, enter :wq!, and press Enter to save the file and exit.
```

Step 4  Run the following command to perform compilation:

```
make LIBS="-lcrypto"
```

Step 5  Run the following commands to load environment variables:

```
export PATH=/path/to/CNVNATOR/CNVnator_v0.4.1/src:$PATH
```

----End

4.10.7 Running and Verifying CNVnator

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to create the working directory:

```
mkdir -p /path/to/CASE
```

Step 3  Download the test case file and use SFTP to upload it to the /path/to/CASE directory.

Step 4  Run the following commands to perform the test:

```
{ time cnvnator -root B17NC.root -tree B17NC.bam ; } 2>&1 |tee -a Mapping.log
{ time cnvnator -root B17NC.root -his 1000 -fasta chr14.GRCm38.primary_assembly.genome.fa.gz ; } 2>&1 |tee -a Histogram.log
```
{ time cnvinator -root B17NC.root -stat 1000 ; } 2>&1 | tee -a Statistics.log

If the command output shown in Figure 4-9 is displayed, the calculation ends normally.

Figure 4-9 Test example

----End

4.10.8 Troubleshooting

Problem 1: "std::unique_ptr" Is Reported During CMake Compilation

Symptom
"std::unique_ptr" is reported during CMake compilation, and the error message is "CMake Error at CMakeLists.txt:92 (message):
The C++ compiler does not support C++11 (e.g. std::unique_ptr)."

Possible Cause
This error is related to the system time setting.

Procedure
Step 1 After the system time is set correctly, decompress the source code package again for compilation.

Or
Run the following command to modify the *CmakeCache.txt* file:

1. `vi CmakeCache.txt`
2. Press `I` to enter the editing mode and modify line 362 in the *CmakeCache.txt* file
   
   ```
   CMake_HAVE_CXX_UNIQUE_PTR:INTERNAL=ON
   ```
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.
4. Perform compilation and installation.
   
   ```make```

----End

**Problem 2: "CMake Error at TBB-stamp" Is Reported During ROOT Compilation**

**Symptom**

"CMake Error at TBB-stamp" is reported during ROOT compilation. The error information is similar to the following:

`CMake Error at TBB-stamp/TBB-download-RelWithDebInfo.cmake:49 (message): Command failed: 1
'/storage/softwares/TaiShan/CNVnator/CMAKE/bin/cmake' '-Dmake=' '-Dconfig=' '-P' '/storage/softwares/TaiShan/CNVnator/ROOT-6.18/TBB-prefix/src/TBB-stamp/TBB-download-RelWithDebInfo-impl.cmake'
```

**Possible Cause**

Such a failure to find the file is due to the cluster's failure to connect to the network.

**Procedure**

Save the downloaded installation packages to the folder generated after CMake is installed. For details, see [Installing CMake](#).

**Problem 3: "[cnvnator] Error 1" Is Reported During CNVnator Compilation**

**Symptom**

"[cnvnator] Error 1" is reported during CNVnator compilation. The error information is similar to the following:

`/usr/bin/ld: samtools/htslib-1.9/libhts.a(hfile_s3.o): undefined reference to symbol 'HMAC@@libcrypto.so.10'
/usr/lib64/libcrypto.so.10: error adding symbols: DSO missing from command line collect2: error: ld returned 1 exit status`

`make: *** [cnvnator] Error 1`

**Possible Cause**

An error occurs during compilation and linking.

**Procedure**
Run the following `make` command:

```
make LIBS="-lcrypto"
```

### 4.10.9 More Information

CNVnator page on GitHub:

[https://github.com/abyzovlab/CNVnator/blob/master/README.md](https://github.com/abyzovlab/CNVnator/blob/master/README.md)

ROOT installation guide:

[https://root.cern.ch/building-root#quick-start](https://root.cern.ch/building-root#quick-start)

### 4.11 Cufflinks 2.2.1 Porting Guide (CentOS 7.6)

#### 4.11.1 Introduction

Cufflinks is a program in the gene sequencing field that is mainly used to calculate the gene expression volume and search for differentially expressed genes. Based on the comparison result of TopHat, the program calculates the FPKM value of isoform (of each gene) based on or not based on the GTF comment file of the reference genome, and provides the comment result in `transcripts.gtf` (assemble transcription groups). Cufflinks contains several main executable programs, such as Cufflinks, Cuffmerge, Cuffcompare, and Cuffdiff.

For more information, visit the [official Cufflinks website](https://cufflinks.cbio.cornell.edu/).

Programming language: C++

Brief description: a program used to calculate the gene expression volume and search for differentially expressed genes.

Open-source protocol: Boost Software License

#### Recommended Version

Cufflinks 2.2.1

#### 4.11.2 Environment Requirements

##### Hardware Requirements

*Table 4-55* lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
### Software Requirements

*Table 4-56* lists the software requirements.

#### Table 4-56 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cufflinks</td>
<td>2.2.1</td>
<td><a href="https://codeload.github.com/cole-trapnell-lab/cufflinks/zip/master">https://codeload.github.com/cole-trapnell-lab/cufflinks/zip/master</a></td>
</tr>
<tr>
<td>Boost</td>
<td>1_55_0</td>
<td><a href="http://downloads.sourceforge.net/project/boost/boost/1.55.0/boost_1_55_0.tar.bz2">http://downloads.sourceforge.net/project/boost/boost/1.55.0/boost_1_55_0.tar.bz2</a></td>
</tr>
<tr>
<td>Eigen</td>
<td>3.3.7</td>
<td><a href="https://gitlab.com/libeigen/eigen/-/archive/3.3.7/eigen-3.3.7.tar.gz">https://gitlab.com/libeigen/eigen/-/archive/3.3.7/eigen-3.3.7.tar.gz</a></td>
</tr>
<tr>
<td>Samtools</td>
<td>0.1.19</td>
<td><a href="https://codeload.github.com/samtools/samtools.tar.gz/0.1.19">https://codeload.github.com/samtools/samtools.tar.gz/0.1.19</a></td>
</tr>
<tr>
<td>htslib</td>
<td>1.9</td>
<td><a href="https://github.com/samtools/htslib/releases/download/1.9/htslib-1.9.tar.bz2">https://github.com/samtools/htslib/releases/download/1.9/htslib-1.9.tar.bz2</a></td>
</tr>
</tbody>
</table>

### OS Requirements

*Table 4-57* lists the OS requirements.

#### Table 4-57 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.11.3 Planning the Paths for Software Porting

*Table Paths for software porting* lists the software installation paths involved in the Cufflinks software porting.

#### Table 4-58 Paths for software porting

<table>
<thead>
<tr>
<th>N o.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the <em>HPC Solution Basic Environment Setup Guide.</em></td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>2</td>
<td><code>/path/to/BOOST</code></td>
<td>Installation path of Boost</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td><code>/path/to/SAMTOOL</code></td>
<td>Installation path of SAMtool</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><code>/path/to/HTSLIB</code></td>
<td>Installation path of HTSlib</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><code>/path/to/EIGEN</code></td>
<td>Installation path of Eigen</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><code>/path/to/CUFFLINKS</code></td>
<td>Installation path of Cufflinks</td>
<td></td>
</tr>
</tbody>
</table>

### 4.11.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Configuration Process

**Table 4-59** Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| 1   | Configure the basic environment. | For details, see the following sections in "Setting Up the Single-Node System Environment" in the *HPC Solution Basic Environment Setup Guide*.  
- Configuring the Local Yum Source  
- Installing GMP  
- Installing MPFR  
- Installing MPC  
- Installing the IB NIC Driver  
- Installing OpenMPI |
| 2   | Install Boost.             | For details, see **4.11.4.1 Installing Boost**. |
| 3   | Install SAMtools.          | For details, see **4.11.4.2 Installing SAMtools**. |
| 4   | Install HTSlib.            | For details, see **4.11.4.3 Installing HTSlib**. |
4.11.4.1 Installing Boost

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies:

```
# NOTE
cufflinks-2.2.1 requires that the Boost version is in the range of 1.47 to 1.55.

yum install -y xz-devel.aarch64 ncurses-devel.aarch64 bzip2-devel.aarch64 libicu.aarch64 libicu-devel.aarch64
```

Step 3 Run the following commands to add the GNU environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 4 Run the following command to decompress the installation package:

```
tar -xvf boost_1_55_0.tar.bz2
```

Step 5 Run the following command to switch to the directory generated after the package is decompressed:

```
cd boost_1_55_0
```

Step 6 Run the following command to run the bootstrap.sh script:

```
./bootstrap.sh
```

Step 7 Run the following commands to compile and install Boost:

```
./b2 --prefix=/path/to/BOOST install
```

Step 8 Ensure that library files are generated in the lib directory of Boost.

```
ls /path/to/BOOST/lib
```

Step 9 Run the following command to set Boost environment variables.

```
export LD_LIBRARY_PATH=/path/to/BOOST/lib:$LD_LIBRARY_PATH
```

----End
4.11.4.2 Installing SAMtools

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to go to the SAMtools installation directory:

```
cd /path/to/SAMTOOLS
```

**Step 3** Run the following command to decompress the SAMtools installation package:

```
tar -xvf samtools-0.1.19.tar.gz
```

**Note**
Install SAMtools and HTSlib separately. Do not use SAMtools that contains HTSlib. Otherwise, errors may occur during compilation.

**Step 4** Run the following command to switch to the directory generated after the package is decompressed:

```
cd samtools-0.1.19
```

**Step 5** Run the following command to obtain the GNU absolute directory:

```
which gcc
```

**Step 6** Modify the `Makefile` file.

```
vi Makefile
```

Before the modification:

```
CC     = gcc
```

After the modification:

```
CC     = /path/to/GNU/bin/gcc
```

**Step 7** Run the following commands to perform compilation and installation:

```
maker
```

**Step 8** Run the following command to copy the generated binary file to the default path of the root directory so that Cufflinks can directly identify the SAMtools and HTSlib files:

```
mkdir /usr/local/include/bam

cp /path/to/SAMTOOLS/samtools-0.1.19/libbam.a /usr/local/lib

cp /path/to/SAMTOOLS/samtools-0.1.19/*.h /usr/local/include/bam/

cp /path/to/SAMTOOLS/samtools-0.1.19/samtools /usr/bin/
```

----End
4.11.4.3 Installing HTSlib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the HTSlib installation directory:

```
cd /path/to/HTSLIB
```

Step 3 Run the following command to decompress the HTSlib installation package:

```
tar -xvf htslib-1.9.tar.bz2
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd htslib-1.9
```

Step 5 Modify the Makefile file.

```
vi Makefile
```

Before the modification:

```
CC     = gcc
```

After the modification:

```
CC     = /path/to/GNU/bin/gcc
```

Step 6 Run the following commands to perform compilation and installation:

```
make
```

Step 7 Run the following command to copy the generated file to the default path of the root directory so that Cufflinks can directly identify the HTSlib files:

```
cp /path/to/HTSLIB/* /usr/local/lib
```

----End

4.11.4.4 Installing Eigen

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the Eigen installation directory:

```
cd /path/to/EIGEN
```

Step 3 Run the following command to decompress the Eigen installation package:

```
tar -xvf eigen-3.3.7.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd eigen-3.3.7
```
**4.11.5 Obtaining the Source Code**

**Procedure**

**Step 1** Download the Cufflinks installation package **cufflinks-master.zip**.
URL: [https://codeload.github.com/cole-trapnell-lab/cufflinks/zip/master](https://codeload.github.com/cole-trapnell-lab/cufflinks/zip/master)

**Step 2** Use SFTP to upload the Cufflinks installation package to the `/path/to/CUFFLINKS` directory on the server.

----End

**4.11.6 Compiling and Installing Cufflinks**

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to go to the Cufflinks installation directory:

```
cd /path/to/CUFFLINKS
```

**Step 3** Run the following command to decompress the Cufflinks installation package:

```
unzip cufflinks-master.zip
```

**Step 4** Run the following command to switch to the directory generated after the package is decompressed:

```
cd cufflinks-master
```

**Step 5** Run the following command to generate a **configure** file:

```
./autogen.sh --install
```

**Step 6** Run the following command perform configuration:

```
./configure --with-boost=/path/to/BOOST --with-eigen=/path/to/EIGEN/eigen-3.3.7 --prefix=/path/to/CUFFLINKS/build
```

**Step 7** Run the following commands to perform compilation and installation:

```
make -j 16 install
```

**Step 8** Run the following commands to add the Cufflinks environment variables:
export PATH=/path/to/CUFFLINKS/build/bin:$PATH

4.11.7 Running and Verifying Cufflinks

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to switch to the test file path of the source code and run the file:

    cd /path/to/CUFFLINKS/cufflinks-master/external_tests/tiny_tests/2x75

Step 3  Run the following commands to convert the case format:

    samtools view -bS accepted_hits.sam > accepted_hits.bam

Step 4  Run the following command to run the case:

    cufflinks accepted_hits.bam

If the following information is displayed, the Cufflinks process is complete:

Warning: Using default Gaussian distribution due to insufficient paired-end reads in open ranges. It is recommended that correct parameters (--frag-len-mean and --frag-dev) be provided.

> Map Properties:
    >       Normalized Map Mass: 90.00
    >       Raw Map Mass: 90.00
    >       Fragment Length Distribution: Truncated Gaussian (default)
    >       Default Mean: 200
    >       Default Std Dev: 80
> Processed 1 loci. [****************************] 100%

4.11.8 Troubleshooting

Problem 1: Error Message "undefined reference to lzma" Is Reported During Cufflinks Compilation

Symptom

An error message "undefined reference to lzma_stream_buffer_bound" is reported when Cufflinks is compiled.

Possible Causes

The LZMA dependency package is not installed.

Procedure

Run the following command to install xz-devel.aarch64:

    yum install xz-devel.aarch64
Problem 2: Error Message "could not detect bam" Is Displayed During Cufflinks Compilation

Symptom
An error message "error:we cand not detect bam"is reported when Cufflinks is compiled.

Possible Causes
- The configure file is incorrectly configured.
- <bam/version.hpp> is used to determine the BAM version in the configure file.

Procedure
- If the configure file is incorrectly configured:
  Run the following commands to modify the configure file:
  a. vi configure
  b. Press i to enter the editing mode and change no to yes in line 5380 of the configure file.
  c. Press Esc, enter :wq!, and press Enter to save the file and exit.
- If <bam/version.hpp> is used to determine the BAM version in the configure file (actually version.h is the version file of the BAM):
  Run the following commands to modify the configure file:
  a. vi configure
  b. Press i to enter the editing mode and change bam/version.hpp to bam/version.h.
  c. Press Esc, enter :wq!, and press Enter to save the file and exit.

Problem 3: Error Message "could not find htslib" Is Displayed During Cufflinks Compilation

Symptom
An error message "could not find htslib"is reported when Cufflinks is compiled.

Possible Causes
The built-in HTSlib of SAMtools is used, which is not supported in the current version.

Procedure
Install HTSlib independently and copy the generated libhts.* file to the lib directory.

Step 1 Run the following commands to install HTSlib:

```
tar -xvf htslib-1.9.tar.bz2
cd htslib-1.9
```
Problem 4: Error Message "GHash.hh:91:44: error" Is Displayed During Cufflinks Compilation

Symptom
An error is reported when Cufflinks is compiled. The error information is as follows:

```
GHash.hh:91:44: error: type/value mismatch at argument 1 in template parameter list for 'template<class _Tp> struct std::hash'
91 |     while (pos<fCapacity && hash[pos].hash<0) pos++;
    |                                            ^
```

Possible Causes
The Cufflinks source code has a bug, which has been fixed in the latest master release.

Procedure
Download the latest Cufflinks installation package.

URL: https://codeload.github.com/cole-trapnell-lab/cufflinks/zip/master

Problem 5: Error Message "error while loading shared libraries" Is Displayed During the Test

Symptom
An error message similar to "error while loading shared libraries: libhts.so.2: cannot open shared object file: No such file or directory" is displayed during the test.

Possible Causes
The HTSlib environment variables are not added.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to specify the HTSlib address:

```
export LD_LIBRARY_PATH=/path/to/HTSLIB/htslib-1.9:SLD_LIBRARY_PATH
```

4.11.9 More Information

Official user guide:

http://cole-trapnell-lab.github.io/cufflinks/getting_started/
4.12 GATK 4.0.0.0 Porting Guide (CentOS 7.6)

4.12.1 Introduction

Genome Analysis Toolkit (GATK), as the name implies, is a toolkit for analyzing genomes. Its main feature is to find variants and genotypes for variant calling from sequencing data, including SNP and INDEL.

GATK includes the following tools:
- Diagnostics and quality control tools
- Sequence data processing tools
- Variant discovery tools
- Variant evaluation tools
- Variant manipulation tools
- Commenting modules
- Reads filtering
- Resource file decoding tools
- Reference sequence utilities

GATK4 is the latest GATK version released by Broad Institute in January 2018. Compared with the previous version, this version has major changes, including newly developed processes and the latest tools based on machine learning algorithms. GATK4 greatly optimizes the performance, computing speed, interface flexibility, and scalability. The end-to-end process can run locally or on the cloud. In addition, it is developed based on Spark and supports cluster deployment.

For more information about GATK, visit the official GATK website.

Programming language: Java

Brief description: A toolkit for analyzing genomes.

Open-source protocol: BSD 3-Clause "New" or "Revised" License

Recommended Version

GATK 4.0.0.0

4.12.2 Environment Requirements

Hardware Requirements

Table 4-60 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
Software Requirements

Table 4-61 lists the software requirements.

Table 4-61 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>GATK</td>
<td>4.0.0.0</td>
<td><a href="https://software.broadinstitute.org/gatk/">https://software.broadinstitute.org/gatk/</a></td>
</tr>
<tr>
<td>Samtools</td>
<td>samtools-0.1.9</td>
<td><a href="https://sourceforge.net/projects/samtools/files/samtools/0.1.9/">https://sourceforge.net/projects/samtools/files/samtools/0.1.9/</a></td>
</tr>
<tr>
<td>blat</td>
<td>35.1</td>
<td><a href="https://codeload.github.com/djhshih/blat.tar.gz/v35.1">https://codeload.github.com/djhshih/blat.tar.gz/v35.1</a></td>
</tr>
<tr>
<td>BWA</td>
<td>0.7.17</td>
<td><a href="https://sourceforge.net/projects/bio-bwa/files/bwa-0.7.17.tar.bz2/download">https://sourceforge.net/projects/bio-bwa/files/bwa-0.7.17.tar.bz2/download</a></td>
</tr>
<tr>
<td>Hadoop</td>
<td>3.1.2</td>
<td><a href="http://mirror.bit.edu.cn/apache/hadoop/common/stable/">http://mirror.bit.edu.cn/apache/hadoop/common/stable/</a></td>
</tr>
<tr>
<td>Spark</td>
<td>2.4.4</td>
<td><a href="https://archive.apache.org/dist/spark/">https://archive.apache.org/dist/spark/</a></td>
</tr>
<tr>
<td>Test case 1</td>
<td>SRR742200_1.fastq</td>
<td>ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR742/SRR742200/SRR742200_1.fastq.gz</td>
</tr>
<tr>
<td>Test case 2</td>
<td>SRR742200_2.fastq</td>
<td>ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR742/SRR742200/SRR742200_2.fastq.gz</td>
</tr>
<tr>
<td>Test case 3</td>
<td>human_g1k_v37.fasta</td>
<td><a href="http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz">http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-62 lists the OS requirements.

Table 4-62 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
4.12.3 Planning the Paths for Software Porting

This chapter lists software installation paths involved in the GATK software porting.

Table 4-63 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/BLAT</td>
<td>Installation path of Blat.</td>
<td>The installation paths listed in this table are only for reference. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/BMA</td>
<td>Installation path of BMA</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/SAMTOOLS</td>
<td>Installation path of SAMtools</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/GATK</td>
<td>Installation path of GATK</td>
<td></td>
</tr>
</tbody>
</table>

4.12.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-64 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>For details, see section &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install dependencies.</td>
<td>For details, see 4.12.4 Configuring the Compilation Environment.</td>
</tr>
<tr>
<td>No.</td>
<td>Operation</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>Install BWA.</td>
<td>For details, see 4.12.4.3 Installing BWA.</td>
</tr>
<tr>
<td>4</td>
<td>Install SAMtools.</td>
<td>For details, see 4.12.4.4 Installing samtools.</td>
</tr>
</tbody>
</table>

4.12.4.1 Installing Dependencies

Procedure

Step 1  Use PuTTY to log in to the server as the root user.
Step 2  Run the following command to install the dependency:

```bash
yum install -y java-1.8.0-openjdk-devel.aarch64 ncurses-devel.aarch64 bzip2-devel.aarch64
```

----End

4.12.4.2 Installing Blat

Procedure

Step 1  Use PuTTY to log in to the server as the root user.
Step 2  Run the following command to install the dependencies:

```bash
yum install libpng-devel -y
```
Step 3  Run the following command to decompress the Blat package:

```bash
tar zxvf blat-35.1.tar.gz
```
Step 4  Run the following command to switch to the directory that contains decompressed files:

```bash
cd blat-35.1/
```
Step 5  Run the following command to declare the installation environment:

```bash
export MACHTYPE=aarch64
```
Step 6  Run the following command to compile and install the software:

```bash
make
```
Step 7  Run the following command to declare the Blat environment variables:

```bash
export PATH=/path/to/BLAT/blat-35.1/bin:$PATH
```

----End
4.12.4.3 Installing BWA

Prerequisites

Obtain the sse2neon.h file from https://github.com/DLTcollab/sse2neon/blob/master/sse2neon.h.

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the BWA installation package.

```
tar -xvf bwa-0.7.17.tar.bz2
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd bwa-0.7.12
```

Step 4 Perform the following steps to modify the Makefile file:

1. `vi Makefile`
2. Press `I` to enter the editing mode and modify lines 1 and 14 in the Makefile file.

   ```
   CC=                     /path/to/GNU/bin/gcc
   INCLUDES=       -I/path/to/GNU/gcc/include
   ```
3. Press `Esc`, type `:wq!`, and press Enter to save the file and exit.

Step 5 Modify the ksw.c file.

1. `vi ksw.c`
2. Press `I` to enter the editing mode and modify lines 29 and 30 in the ksw.c file.

   ```
   /*#include <emmintrin.h>*/
   #include <sse2neon.h>
   ```
3. Press `Esc`, type `:wq!`, and press Enter to save the file and exit.

Step 6 Copy the Neon instruction conversion code to a specific directory of the compiler.

```
cp sse2neon.h /path/to/GNU/gcc/include
```

Step 7 Run the following commands to perform compilation:

```
make
```

Step 8 Run the following commands to set the BWA environment variables:

```
export PATH=/path/to/BMA/bwa-0.7.12:$PATH
```

Step 9 Run the following command to check whether the installation is successful:

```
bwa
```

If information similar to the following is displayed, the installation is successful:

```
bwa <command> [options]
```
4.12.4.4 Installing samtools

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to decompress the SAMtools installation package:

```
tar xvjf samtools-0.1.2.tar.bz2
```

Step 3  Run the following command to switch to the directory generated after decompression:

```
cd bwa-0.7.12
```

Step 4  Run the following command to install the dependency:

```
yum install xz-devel-5.2.2-1.el7.aarch64 bzip2-devel-1.0.6-13.el7.aarch64
ncurses-devel.aarch64 -y
```

Step 5  Run the following commands to perform the configuration:

```
./configure --prefix=/path/to/SAMTOOLS
```

Step 6  Run the following commands to perform compilation:

```
make
make install
```

Step 7  Run the following commands to set the SAMtools environment variables:

```
export PATH=/path/to/SAMTOOLS/samtools-1.9/bin:$PATH
```

Step 8  Run the following command to check whether the installation is successful:

```
samtools
```

If information similar to the following is displayed, the installation is successful:

<table>
<thead>
<tr>
<th>Program: samtools  (Tools for alignments in the SAM format)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version: 1.9 (using htslib 1.9)</td>
</tr>
<tr>
<td>Usage: samtools &lt;command&gt; [options]</td>
</tr>
</tbody>
</table>

----End

4.12.5 Obtaining the Source Code

Procedure

Step 1  Download the GATK installation package gatk-4.0.0.0.zip.

```
https://software.broadinstitute.org/gatk/
```

Step 2  Use the SFTP tool to upload the GATK installation package to the /path/to/GATK directory on the server.

----End
4.12.6 Compiling and Installing GATK

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to install GATK4.

unzip gatk-4.0.0.0.zip
export PATH=/path/to/GATK/gatk-4.0.0.0:$PATH
gatk –help

Step 3  Start the Hadoop-Spark service.

For details about how to install and deploy the Hadoop and Spark, see Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6).

----End

4.12.7 Running and Verifying GATK

4.12.7.1 Running GATK4 in Spark Mode

Prerequisites

Prepare data files. Three types of data files are required for the GATK test: genome data files (in FASTA format), original sequencing data files (FASTQ or BAM format), and target area BED files.

This test case uses the following data files:

1. Genome data files (in FASTA format)
   The following is the human genome information.
   - human_g1k_v37.fasta
   The following two VCF files are from the 1000 Genomes Project and the Mills Project, which record the population InDel areas detected in those projects.
   - Mills_and_1000G_gold_standard.indels.hg19.sites.vcf.gz
   - 1000G_phase1.indels.hg19.sites.vcf.gz
   The following document is a collection of almost all public population variants currently available.
   - dbsnp132_20101103.vcf
   The mutation dataset selects one to three types of data based on the sequencing purpose.

2. Original sequencing data (FASTQ or BAM format)
   - SRR742200_1.fastq
   - SRR742200_2.fastq
   a. Use PuTTY to log in to the server as the root user.
   b. Run the following command to create an input folder:
mkdir projectDir

cd projectDir

mkdir input

c. Run the following commands to decompress the case files:

gzip -d SRR742200_1.fastq.gz

gzip -d SRR742200_1.fastq.gz

gzip -d human_g1k_v37.fasta.gz

gzip -d dbsnp132_20101103.vcf.gz

d. Run the following command to generate a 2-bit file using Blat:

faToTwoBit human_g1k_v37.fasta human_g1k_v37.fasta.2bit

e. Run the following command to save all data files to the input folder:

cp human_g1k_v37.fasta human_g1k_v37.fasta.2bit
SRR742200_1.fastq SRR742200_2.fastq
dbsnp132_20101103.vcf input

3. Target area bed file.
   - illumina_pt2.bed

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to create an index for the FASTA file:

bwa index -a bwtsw human_g1k_v37.fasta

The command output is as follows:

[bwa_index] Pack FASTA... 27.89 sec
[bwa_index] Construct BWT for the packed sequence...
[BWTincCreate] textLength=6203609478, availableWord=448508744
[BWTincConstructFromPacked] 10 iterations done. 99999990 characters processed.
[BWTincConstructFromPacked] 20 iterations done. 199999990 characters processed.
[BWTincConstructFromPacked] 30 iterations done. 299999990 characters processed.
[BWTincConstructFromPacked] 40 iterations done. 399999990 characters processed.
[BWTincConstructFromPacked] 50 iterations done. 499999990 characters processed.
[BWTincConstructFromPacked] 60 iterations done. 599999990 characters processed.
[BWTincConstructFromPacked] 70 iterations done. 699999990 characters processed.
[BWTincConstructFromPacked] 80 iterations done. 799999990 characters processed.
[BWTincConstructFromPacked] 90 iterations done. 899999990 characters processed.
<The following output is omitted.>

Step 3  Run the following command to view the generated index files:

ls

□□ NOTE

After steps 2 and 3 are performed, the human_g1k_v37.fasta.amb, human_g1k_v37.fasta.ann, human_g1k_v37.fasta.bwt, human_g1k_v37.fasta.pac, and human_g1k_v37.fasta.sa files are displayed in the current path. The generated index files are common to all pipelines and need to be constructed only once. It takes about one and a half hours. The generated files will be used in subsequent sequencing steps.

Step 4  BWA mapping compares the FASTQ files.

1. Run the following command to generate a dictionary:

gatk CreateSequenceDictionary -R human_g1k_v37.fasta -O human_g1k_v37.dict
If the information in the red box shown in the following figure is displayed, the execution is complete.

The generated dict file is used for genome comparison.

2. Run the following commands to perform comparison:

```
bwa mem -M -t 96 human_g1k_v37.fasta SRR742200_1.fastq SRR742200_2.fastq > SRR7.sam
```

<table>
<thead>
<tr>
<th>Table 4-65 Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>-t</td>
</tr>
</tbody>
</table>

If the information in the red box shown in the following figure is displayed, the execution is complete.

Step 5  Run the following command to re-sort the gene sequence:

By default, the sam files generated by the BWA are sorted by dictionary mode. You need to convert the files for sorting by genome type, which is completed by the Picard tool integrated in GATK4.

```
gatk ReorderSam -I SRR7.sam -O SRR7_reorder.bam -R human_g1k_v37.fasta
```

If the information in the red box shown in the following figure is displayed, the execution is complete.

Step 6  Sort gene sequences by coordinate in descending order.
1. Spark runs in the HDFS file system and is invisible in the Linux system. Run the following command to transfer the SRR7_reorder.bam file to the HDFS file system:

   hadoop fs -mkdir /seqdata
   hadoop fs -put SRR7_reorder.bam /seqdata

2. Run the following command to check whether the sparklog directory has been manually created:

   hadoop fs -mkdir /sparklog
   hadoop fs -ls /

3. Run the following commands to perform sorting:

   ```bash
   ```

   **NOTE**
   
   Change the host name based on the master node deployed for Hadoop and Spark. For details, see Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6).

   ![Red box from figure]
   
   If the information in the red box shown in the following figure is displayed, the execution is complete.

4. Run the following command to copy the generated file to the data directory:

   ```bash
   hadoop fs -get /seqdata/SRR7_sorted.bam ./
   ```

5. Run the following command to view the data directory:

   ```bash
   ll
   ```

   If the information in the red box shown in the following figure is displayed, the execution is complete.
Step 7  Run the following command to add a head for the gene sequence.

GATK 2.0 and later versions do not support headless file detection. This step can be performed by using -r during bwa comparison or by using the AddOrReplaceReadGroups tool in GATK4.

```bash
gatk AddOrReplaceReadGroups -I SRR7_sorted.bam -O SRR7_header.bam -LB lib1 -PL illumina -PU unit1 -SM 20
```

If the information in the red box shown in the following figure is displayed, the execution is complete.

Step 8  Deduplicate gene sequences.

1. Run the following commands to upload the head file to HDFS:

   ```bash
   hadoop fs -put SRR7_header.bam /seqdata
   ```

2. Run the following command to query HDFS:

   ```bash
   hadoop fs –ls /seqdata
   ```

3. Run the following command to perform deduplication.

   ```bash
   ```

   **NOTE**
   Change the host name based on the master node deployed for Hadoop and Spark. For details, see Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6).
Step 9 The BQSR performs recalibration.

1. Run the following commands to upload the original file to HDFS:
   ```bash
   hadoop fs -put dbsnp132_20101103.vcf /seqdata
   hadoop fs -put human_g1k_v37.fasta.2bit /seqdata
   hadoop fs -ls /seqdata
   ```
   If the information in the red box shown in the following figure is displayed, the execution is complete.

2. Perform base quality score recalibration (BQSR).
   ```bash
   gatk BQSRPipelineSpark -I hdfs://Hostname:9000/seqdata/
   SRR7_markdup.bam -R hdfs://Hostname:9000/seqdata/
   human_g1k_v37.fasta.2bit -O hdfs://Hostname:9000/seqdata/
   SRR7_bqsr.bam --known-sites hdfs://Hostname:9000/seqdata/
   dbsnp132_20101103.vcf --disable-sequence-dictionary-validation true -- --
   spark-runner SPARK --spark-master spark://Hostname:7077
   ```
   **NOTE**
   Change the host name based on the master node deployed for Hadoop and Spark. For details, see Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6) Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6).
Step 10  Run the following commands to check the HaplotypeCaller mutation process.

Detect mutations and generate a variant vcf file.

```
```

**NOTE**

Change the host name based on the master node deployed for Hadoop and Spark. For details, see Hadoop 3.1.2 + Spark 2.4.4 Porting Guide (CentOS 7.6).
4.12.7.2 Running GATK4 in Non-Spark Mode

Procedure

**Step 1** Use PuTTY to log in to the server as the `root` user.

**Step 2** Run the following command to create an index for the FASTA file:

```
bwa index -a bwtsw human_g1k_v37.fasta
```

**Step 3** Run the following commands to compare the FASTQ files:

```
gatk CreateSequenceDictionary -R human_g1k_v37.fasta -O human_g1k_v37.dict
bwa mem -M -t 64 human_g1k_v37.fasta SRR742200_1.fastq SRR742200_2.fastq > SRR7.sam
```

**Step 4** Run the following command to re-sort the gene sequence:

```
gatk ReorderSam -I SRR7.sam -O SRR7_reorder.bam -R human_g1k_v37.fasta
```
Step 5 Run the following command to sort gene sequences by coordinate in descending order.

`gatk SortSam -I SRR7_reorder.bam -O SRR7_sorted.bam --SORT_ORDER coordinate`

Step 6 Run the following command to add a head for the gene sequence.

`gatk AddOrReplaceReadGroups -I SRR7_sorted.bam -O SRR7_addhead.bam -LB lib1 -PL illumina -PU unit1 -SM 20`

Step 7 Run the following command to deduplicate the gene sequences.

`gatk MarkDuplicates -I SRR7_addhead.bam -M test.metric -O SRR7_markdup.bam`

Step 8 Run the following command to perform recalibration.

1. Run the following command to re-sort the FASTA reference files to generate the fai file:

   `samtools faidx human_g1k_v37.fasta > human_g1k_v37.fai`

2. Run the following commands to index the VCF file:

   `gatk IndexFeatureFile -F dbsnp132_20101103.vcf`
   `gatk BaseRecalibrator -I SRR7_markdup.bam --known-sites dbsnp132_20101103.vcf -O SRR7_bqsr.bam -R human_g1k_v37.fasta`
   `gatk ApplyBQSR -bqsr SRR7_bqsr.bam -I SRR7_markdup.bam -O SRR7_aybqsr.bam`

Step 9 Run the following commands to check the HaplortypeCaller mutation process.

`gatk HaplotypeCaller -I SRR7_aybqsr.bam -O SRR7_raw.vcf -R human_g1k_v37.fasta`

----End

4.12.8 Troubleshooting

An Error Is Reported When GATK Is Run

**Symptom:**

An error message similar to the following is displayed when GATK is run: Unable to load libgkl_compression.so from native/libgkl_compression.so (/tmp/libgkl_compression8107837509941713683.so: /tmp/libgkl_compression8107837509941713683.so: cannot open shared object file: No such file or directory (Possible cause: can't load AMD 64-bit .so on a AARCH64-bit platform)).

**Possible Cause**

GKL is an acceleration library developed based on the Intel CPU, and needs to be ported to the Kunpeng platform.

**Procedure**

Ignore this error if performance comparison is not involved.
4.12.9 More Information

GATK user guide
https://software.broadinstitute.org/gatk/documentation/

Gene file database
https://hgdownload.soe.ucsc.edu/downloads.html

4.13 HISAT2 2.1.0 Porting Guide (CentOS 7.6)

4.13.1 Introduction

HISAT2 is a fast and sensitive alignment program for mapping next-generation sequencing reads (whole-genome, transcriptome, and exome sequencing data) against the general human population (as well as against a single reference genome). Based on an extension of BWT for a graph, a graph FM index (GFM) has been designed and implemented. In addition to using one global GFM index that represents general population, HISAT2 uses a large set of small GFM indexes that collectively cover the whole genome (each index representing a genomic region of 56 kbp, with 55,000 indexes needed to cover human population).

For more information about HISAT2, visit the official HISAT2 website.

Programming language: C++

Brief description: fast and sensitive alignment program.

Open-source license: GPL 3.0

Recommended Version

HISAT2 2.1.0

4.13.2 Environment Requirements

Hardware Requirements

Table 4-66 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-67 lists the software requirements.
### Table 4-67 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>HISAT2</td>
<td>2.1.0</td>
<td><a href="https://github.com/infphilo/hisat2">https://github.com/infphilo/hisat2</a></td>
</tr>
<tr>
<td>FASTQ file</td>
<td>R1.fq.gz</td>
<td><a href="http://opengene.org/data/R1.fq.gz">http://opengene.org/data/R1.fq.gz</a></td>
</tr>
<tr>
<td>FASTQ file</td>
<td>R2.fq.gz</td>
<td><a href="http://opengene.org/data/R2.fq.gz">http://opengene.org/data/R2.fq.gz</a></td>
</tr>
</tbody>
</table>

### OS Requirements

**Table 4-68** lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.13.3 Planning the Paths for Software Porting

**Table 4-69** lists the software installation paths involved in the HISAT2 software porting.
Table 4-69 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package for setting up the basic environment</td>
<td>For details, see section Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HISAT2</td>
<td>Installation path of HISAT2</td>
<td>The installation paths listed in this table are only an example. Shared paths are recommended. The actual installation paths may be different, and you need to change the installation paths in subsequent commands in this document based on the actual situation.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/CASE</td>
<td>Planned path for storing the test cases of HISAT2</td>
<td></td>
</tr>
</tbody>
</table>

4.13.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-70 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>For details, see section &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

4.13.5 Obtaining the Source Code

Procedure

**Step 1** Download the HISAT2 installation package `hisat2-2.1.0.tar.gz`.  
URL: [https://github.com/infphilo/hisat2](https://github.com/infphilo/hisat2)
4.13.6 Compiling and Installing HISAT2

Prerequisites


Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Copy the downloaded `sse2neon.h` file to the `/path/to/GNU/lib/gcc/aarch64-unknown-linux-gnu/9.1.0/include` directory.

**Step 3** Run the following command to decompress the HISAT2 installation package:

```bash
cd /path/to/HISAT2

tar -xvf hisat2-2.1.0.tar.gz

cd hisat2-2.1.0
```

**Step 4** Run the following commands to set environment variables:

```bash
export NO_TBB=1
export POPCNT_CAPABILITY=0
```

**Step 5** Perform the following steps to modify the **Makefile** file:

1. Run the `vi` command to open the **Makefile** file.

   ```bash
   vi Makefile
   ```

2. Press `i` to go to the insert mode, and modify the **Makefile** file by inserting the content below in bold between line 140 and line 141:

   Currently, ”BITS=64” supports only x86 and does not support aarch64. The aarch64 branch needs to be added.

   **Before the modification:**
   ```make
   BITS=32
   ifeq (x86_64,$(shell uname -m))
   BITS=64
   endif
   ```

   **After the modification:**
   ```make
   BITS=32
   ifeq (x86_64,$(shell uname -m))
   BITS=64
   endif
   ```

   ```make
   ifeq (aarch64,$(shell uname -m))
   BITS=64
   endif
   ```

3. Press **Esc**, type `:wq!`, and press **Enter** to save the file and exit.

**Step 6** Perform the following steps to modify the **Makefile** file:
1. Run the `vi` command to open the Makefile file.

   `vi Makefile`

2. Press `i` to go to the insert mode, and modify the Makefile file by inserting the content below in bold between line 159 and line 160.

   The compilation options `-m64` and `-msse2` in Makefile supports only x86 and does not support aarch64. The aarch64 branch needs to be added.

   Before the modification (line 149 to line 156 in the source file):
   ```
   ifeq (32,$(BITS))
   BITS_FLAG = -m32  
   endif
   ifeq (64,$(BITS))
   BITS_FLAG = -m64
   endif
   SSE_FLAG=-msse2
   ```

   After the modification:
   ```
   ifeq (32,$(BITS))
   BITS_FLAG = -m32
   endif
   ifeq (64,$(BITS))
   BITS_FLAG = -m64
   endif
   SSE_FLAG=-msse2
   ifeq (aarch64,$(shell uname -m))
   BITS_FLAG =
   SSE_FLAG =
   endif
   ```

3. Press Esc, type `:wq!`, and press Enter to save the file and exit.

   **Step 7** Perform the following steps to modify the Makefile file:

   1. Run the `vi` command to open the Makefile file.

      `vi Makefile`

   2. Press `i` to go to the insert mode and modify the `FLAGS` parameters in the Makefile file

      Before the modification:
      ```
      EXTRA_FLAGS += -DPOPCNT_CAPABILITY
      DEBUG_FLAGS = -00 -g3 $(BIToS_FLAG) $(SSE_FLAG)
      DEBUG_DEFS = -DCOMPILER_OPTIONS="\"$(DEBUG_FLAGS) $(EXTRA_FLAGS)\""
      RELEASE_FLAGS = -O3 $(BITS_FLAG) $(SSE_FLAG) -funroll-loops -g3
      RELEASE_DEFS = -DCOMPILER_OPTIONS="\"$(RELEASE_FLAGS) $(EXTRA_FLAGS)\""
      NOASSERT_FLAGS = -D NDEBUG
      FILE_FLAGS = -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE
      ```

      After the modification:
      ```
      EXTRA_FLAGS +=
      DEBUG_FLAGS = -00 -g3 $(BIToS_FLAG) $(SSE_FLAG) -funroll-loops -std=c++98
      DEBUG_DEFS = -DCOMPILER_OPTIONS="\"$(DEBUG_FLAGS) $(EXTRA_FLAGS)\"
      RELEASE_FLAGS = -O3 $(BITS_FLAG) $(SSE_FLAG) -funroll-loops -g3 -std=c++98 -X linker --allow-multiple-definition
      RELEASE_DEFS = -DCOMPILER_OPTIONS="\"$(RELEASE_FLAGS) $(EXTRA_FLAGS)\""
      NOASSERT_FLAGS = -DNDEBUG
      FILE_FLAGS = -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE
      ```

3. Press Esc, type `:wq!`, and press Enter to save the file and exit.

   **Step 8** Perform the following steps to modify the aligner_sw.h and sse_util.h header files:

   1. Upload the `sse2neon.h` file to the third_party directory.
2. Run the vi command to open the `aligner_sw.h` file.
   
   ```bash
   vi aligner_sw.h
   ```

3. Press `i` to go to the edit mode. Modify the `aligner_sw.h` file by replacing 
   ```c
   #include <emmintrin.h>
   ```
   with 
   ```c
   #include <sse2neon.h>
   ```

   Before the modification:
   ```c
   #include <emmintrin.h>
   ```

   After the modification:
   ```c
   #include <sse2neon.h>
   ```

4. Press `Esc`, type `:wq!`, and press `Enter` to save the file and exit.
5. Modify the `sse_util.h` file in the same way.

Step 9 Run the following commands to compile and install HISAT2.

```bash
make
```

----End

4.13.7 Running and Verifying HISAT2

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Go to the directory where the test case is stored.

**Step 3** Run the following command to go to the `/path/to/CASE` directory:

```bash
cd /path/to/CASE
```

**Step 4** Run the following command to set the environment variables for the HISAT executable file:

```bash
export PATH=/path/to/HISAT2/hisat2-2.1.0/:$PATH
```

**Step 5** Run the following command to decompress the `bdgp6.tar.gz` package in the Software list:

```bash
cd /path/to/CASE

tar -xvf bdgp6.tar.gz

cd bdgp6
```

**Step 6** Perform the following steps to modify the `make_bdgp6.sh` file:

1. Run the vi command to open the `make_bdgp6.sh` file.

   ```bash
   vi make_bdgp6.sh
   ```

2. Press `i` to go to the insert mode and modify the `make_bdgp6.sh` file.

   Comment out the following line:
   ```c
   # ENSEMBL_BASE=ftp://xxxxxxxx
   ```

3. Press `Esc`, type `:wq!`, and press `Enter` to save the file and exit.

**Step 7** Use an SFTP tool to copy the

`Drosophila_melanogaster.BDGP6.dna.toplevel.fa.gz` file to the `/path/to/CASE/bdgp6` directory.
Step 8 Run the following commands to decompress the files:

```
gzip -d Homo_sapiens.GRCh38.86.chr.gtf.gz
gzip -d R1.fq.gz
gzip -d R2.fq.gz
```

Step 9 Execute the following command to run the `make_bdgp6.sh` file:

```
./make_bdgp6.sh
```

Step 10 Run the following commands to create the transcriptome.

```
extract_exons.py Homo_sapiens.GRCh38.86.chr.gtf > genome.exon
extract_splice_sites.py Homo_sapiens.GRCh38.86.chr.gtf > genome.ss
```

Step 11 Run the following command to index the genome and transcriptome:

```
hisat2-build -p96 genome.fa --ss genome.ss --exon genome.exon genome_tran
```

Step 12 Execute the following command to run HISAT2:

```
hisat2 -t -p 96 -x genome_tran -1 R1.fq -2 R2.fq -S Test.sam
```

Check the value of *Overall time* (the format is [hh]:[mm]:[ss]) in the log. A smaller value indicates better performance.

*Figure 4-10* shows an example of the test result.

---

4.13.8 More Information

daehwankimlab website:


4.14 STAR 2.7.1a Porting Guide (CentOS 7.6)
4.14.1 Introduction

STAR is an ultrafast universal RNA-seq aligner.

Motivation

Accurate alignment of high-throughput RNA-seq data is a challenging and yet unsolved problem because of the non-contiguous transcript structure, relatively short read lengths and constantly increasing throughput of the sequencing technologies. Currently available RNA-seq aligners suffer from high mapping error rates, low mapping speed, read length limitation and mapping biases.

Results

To align our large (>80 billion reads) ENCODE Transcriptome RNA-seq dataset, we developed the Spliced Transcripts Alignment to a Reference (STAR) software based on a previously undescribed RNA-seq alignment algorithm that uses sequential maximum mappable seed search in uncompressed suffix arrays followed by seed clustering and stitching procedure. STAR outperforms other aligners by a factor of >50 in mapping speed, aligning to the human genome 550 million 2 × 76 bp paired-end reads per hour on a modest 12-core server, while at the same time improving alignment sensitivity and precision. In addition to unbiased de novo detection of canonical junctions, STAR can discover non-canonical splices and chimeric (fusion) transcripts, and is also capable of mapping full-length RNA sequences. Using Roche 454 sequencing of reverse transcription polymerase chain reaction amplicons, we experimentally validated 1960 novel intergenic splice junctions with an 80–90% success rate, corroborating the high precision of the STAR mapping strategy.

Availability and Implementation

STAR is implemented as a standalone C++ code. STAR is free open source software distributed under GPLv3 license and can be downloaded from http://code.google.com/p/rna-star/.

For more information, visit the official STAR website.

Programming language: C++

Brief description: an RNA-seq aligner

Open-source license: GPL 3.0

Recommended Version

STAR 2.7.1a

4.14.2 Environment Requirements

Hardware Requirements

Table 4-71 lists the hardware requirements.
Table 4-71 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-72 lists the software requirements.

Table 4-72 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAR</td>
<td>2.7.1a</td>
<td><a href="https://github.com/alexdobin/STAR">https://github.com/alexdobin/STAR</a></td>
</tr>
<tr>
<td>Test case file 1</td>
<td>Homo_sapiens.GRC h38.dna.chromosome.2.fa.gz</td>
<td>ftp://ftp.ensembl.org/pub/release-97/fasta/homo_sapiens/dna/</td>
</tr>
<tr>
<td>Test case file 3</td>
<td>TG_r1.fastq.gz</td>
<td><a href="http://tools.genxpro.net/omiras/testdata/">http://tools.genxpro.net/omiras/testdata/</a></td>
</tr>
<tr>
<td></td>
<td>TG_r2.fastq.gz</td>
<td></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-73 lists the OS requirements.

Table 4-73 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.14.3 Planning the Paths for Software Porting

Table 4-74 lists the software installation paths involved in the STAR software porting.
### Table 4-74 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the <em>HPC Solution Basic Environment Setup Guide.</em></td>
</tr>
<tr>
<td>2</td>
<td>/path/to/STAR</td>
<td>Installation path of STAR</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/CASE</td>
<td>Path for storing the STAR test cases</td>
<td></td>
</tr>
</tbody>
</table>

### 4.14.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.

**Configuration Process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the <em>HPC Solution Basic Environment Setup Guide.</em></td>
</tr>
</tbody>
</table>

### 4.14.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the STAR installation package STAR-2.7.1a.tar.gz.

URL: [https://github.com/alexdobin/STAR](https://github.com/alexdobin/STAR)

**Step 2** Use SFTP to upload the STAR installation package to the /path/to/STAR directory on the server.

----End
4.14.6 Compiling and Installing STAR

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Install the zlib-devel package.

```bash
yum install zlib-devel -y
```

**NOTE**

In this example, the CentOS 7.6 image is used as the local YUM source. You can change the YUM source in the `/etc/yum.repos.d` directory as required.

Step 3 Run the following commands to decompress the STAR installation package:

```bash
tar -xvf STAR-2.7.1a.tar.gz
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd STAR-2.7.1a
export BLAST_TOP=`pwd`
```

Step 5 Run the following commands to modify the Makefile file:

1. `vi source/Makefile`
2. Press i to enter the editing mode and edit the Makefile file.
   ```bash
   LDFLAGSextra ?= -flto
   CXXFLAGSextra ?= -flto
   -march=native -mcpu=tsv110
   LSFLAGSextra ?= -flto
   LDFLAGSextra ?= -flto
   -march=native -mcpu=tsv110
   
   export BLAST_TOP="pwd"
   ```
3. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 6 Run the following commands to compile and install STAR:

```bash
cd source
make STAR
```

Step 7 Run the following command to check whether an executable file is generated:

```bash
ll /path/to/STAR/STAR-2.7.1a/source/STAR
```

Step 8 Run the following commands to set STAR environment variables:

```bash
export PATH=/path/of/STAR/STAR-2.7.1a/source/:SPATH
```

----End

4.14.7 Running and Verifying STAR

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to switch to the directory where the test cases are stored:

```bash
```
cd /path/to/CASE

**Step 3** Run the following commands to decompress the test case packages:

```bash
gzip -d Homo_sapiens.GRCh38.dna.chromosome.2.fa.gz

gzip -d Homo_sapiens.GRCh38.86.chr.gtf.gz
```

**Step 4** Run the following commands to create a genome index:

```bash
mkdir chr1_hg38_index

{ time STAR --runThreadN 96 --genomeSAindexNbases 12 --runMode genomeGenerate --genomeDir chr1_hg38_index --genomeFastaFiles Homo_sapiens.GRCh38.dna.chromosome.2.fa --sjdbGTFfile Homo_sapiens.GRCh38.86.chr.gtf --sjdbOverhang 99 ; } 2>&1 | tee genome-index.log
```

**Step 5** Run the following command to compare genomes:

```bash
{ time STAR --runMode alignReads --outSAMtype BAM Unsorted --readFilesCommand zcat --genomeDir chr1_hg38_index/ --outFileNamePrefix Homo_sapiens.GRCh38 --readFilesIn TG_r1.fastq.gz TG_r2.fastq.gz ; } 2>&1 | tee Mapping.log
```

**Step 6** After the genome comparison is complete, the following files and directories are generated:

- Homo_sapiens.GRCh38Aligned.out.bam
- Homo_sapiens.GRCh38Log.final.out
- Homo_sapiens.GRCh38Log.out
- Homo_sapiens.GRCh38Log.progress.out
- Homo_sapiens.GRCh38SJ.out.tab
- Homo_sapiens.GRCh38_STARtmp

**Step 7** When the genome index is created and genome comparison is performed, the software does not automatically generate logs about the running time or performance. You need to run the `time` command in the test command to redirect the entire running time to the log file.

View the **real** value in the last `time` command output in the log. The value indicates \( x \) minutes \( y \) seconds. The lower the value is, the better the performance is.

Information similar to the following is displayed.

```
[root@XA320V2-19 CASE]# cat Mapping.log
Jan 16 20:42:41 ..... started STAR run
Jan 16 20:42:41 ..... loading genome
Jan 16 20:42:42 ..... started mapping
Jan 16 20:42:53 ..... finished mapping
Jan 16 20:42:53 ..... finished successfully
real    0m11.581s
user    0m10.703s
sys     0m0.579s
```

-----End
4.14.8 More Information

HBCTraining website:

https://hbctraining.github.io/Intro-to-rnaseq-hpc-O2/lessons/03_alignment.html

4.15 Tophat2 2.1.1 Porting Guide (CentOS 7.6)

4.15.1 Introduction

TopHat2 is a program that quickly splices and maps RNA-Seq data. It uses the ultra-fast high-throughput short-read comparison program to compare RNA-Seq information to the mammalian size genome, then the mapping results are analyzed to identify the splice points between exons. TopHat is the result of a joint development of the University of Maryland Center for Bioinformation and Computer Biology, the Departments of Mathematics and Molecular and Cell Biology of the University of California, Berkeley, and the Harvard Department of Stem Cell and Regenerative Biology.

For more information about TopHat2, visit the official TopHat2 website.

Programming language: C

Brief description: a program that quickly splices and maps RNA-Seq data

Open-source protocol: Boost Software License 1.0

Recommended Version

TopHat2 2.1.1

4.15.2 Environment Requirements

Hardware Requirements

Table 4-76 lists the hardware requirements.

Table 4-76: Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-77 lists the software requirements.
Table 4-77 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>TopHat2</td>
<td>2.1.1</td>
<td><a href="http://ccb.jhu.edu/software/tophat/downloads/tophat-2.1.1.tar.gz">http://ccb.jhu.edu/software/tophat/downloads/tophat-2.1.1.tar.gz</a></td>
</tr>
<tr>
<td>Bowtie2</td>
<td>2.3.5.1</td>
<td><a href="https://netix.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.3.5.1/bowtie2-2.3.5.1-source.zip">https://netix.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.3.5.1/bowtie2-2.3.5.1-source.zip</a></td>
</tr>
<tr>
<td>SIMDe</td>
<td>master</td>
<td><a href="https://github.com/nemequ/simde">https://github.com/nemequ/simde</a></td>
</tr>
<tr>
<td>Boost</td>
<td>1.70.0</td>
<td><a href="https://dl.bintray.com/boostorg/release/1.70.0/source/">https://dl.bintray.com/boostorg/release/1.70.0/source/</a></td>
</tr>
<tr>
<td>Test case</td>
<td>N/A</td>
<td><a href="http://ccb.jhu.edu/software/tophat/downloads/test_data.tar.gz">http://ccb.jhu.edu/software/tophat/downloads/test_data.tar.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-78 lists the OS requirements.

Table 4-78 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.15.3 Planning the Paths for Software Porting

Table 4-79 lists the software installation paths involved in the TopHat2 2.1.1 software porting.

Table 4-79 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/BOWTIE2</td>
<td>Installation path of Bowtie2</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in</td>
</tr>
</tbody>
</table>
### 4.15.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Configuration Process

**Table 4-80 Configuration process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see “Setting Up the Cluster Environment” in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>Install Bowtie2.</td>
<td>For details, see <a href="#">4.15.4.1 Installing bowtie2</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Install Boost.</td>
<td>For details, see <a href="#">4.15.4.2 Installing boost</a>.</td>
</tr>
</tbody>
</table>

**4.15.4.1 Installing bowtie2**

The Bowtie2 or Bowtie commands are required for TopHat data test in the PATH environment. Therefore, you need to install Bowtie2 first.

#### Procedure

**Step 1** Use PuTTY to log in to the server as the `root` user.

**Step 2** Run the following command to switch to the Bowtie2 installation directory:

```
cd /path/to/BOWTIE2
```

**Step 3** Run the following command to decompress the Bowtie2 source code package:

```
unzip bowtie2-2.3.5.1-source.zip
```
Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd bowtie2-2.3.5.1
```

Step 5 Run the following command to decompress SIMDe:

```bash
unzip simde-master.zip
```

Step 6 Run the following command to copy the SIMDe files to Bowtie2:

```bash
cp -a simde-master/simde bowtie2-2.3.5.1/third_party/simde/
```

Step 7 Run the following command to check whether the `bowtie2-2.3.5.1/third_party/simde/simde/x86/sse2.h` file exists:

```bash
cd /path/to/BOWTIE2/bowtie2-2.3.5.1/third_party/simde/simde/x86
ll sse2.h
```

```
-rw-r--r-- 1 root root 99060 Aug 5 04:12 sse2.h
```

Step 8 Run the following command to switch to the `bowtie2-2.3.5.1` directory:

```bash
cd /path/to/BOWTIE2/bowtie2-2.3.5.1
```

Step 9 Run the following command to perform compilation:

```bash
make NO_TBB=1 POPCNT_CAPABILITY=0 all
```

Step 10 Run the following command to load environment variables:

```bash
export PATH=/path/to/BOWTIE2/bowtie2-2.3.5.1:SPATH
```

-----End

## 4.15.4.2 Installing boost

### Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to decompress the Boost installation package:

```bash
tar xvf boost_1_70_0.tar.bz2
```

Step 3 Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd boost_1_70_0
```

Step 4 Because TopHat requires only the Boost thread and system libraries, Bootstrap needs to contain only the two libraries. Use `--prefix` to specify the Boost installation path.

```bash
./bootstrap.sh --with-libraries=system,thread --prefix=/path/to/BOOST --with-toolset=gcc
```

Step 5 Run the following commands to perform compilation and installation:

```bash
./b2 -d+2 -q -j 32 threading=multi link=shared variant=release --prefix=/path/to/BOOST install
```
Step 6  Run the following command to load environment variables:

```
export LD_LIBRARY_PATH=/path/to/BOOST/lib:SLD_LIBRARY_PATH
```

---End

### 4.15.5 Obtaining the Source Code

**Procedure**

**Step 1**  Download the installation package `tophat-2.1.1.tar.gz`.

URL: [http://ccb.jhu.edu/software/tophat/downloads/tophat-2.1.1.tar.gz](http://ccb.jhu.edu/software/tophat/downloads/tophat-2.1.1.tar.gz)

**Step 2**  Use SFTP to upload the TopHat2 installation package to the `/path/to/TOPHAT2` directory on the server.

---End

### 4.15.6 Compiling and Installing TopHat2

**Procedure**

**Step 1**  Use PuTTY to log in to the server as the root user.

**Step 2**  Run the following command to go to the `TOPHAT2` directory:

```
cd /path/to/TOPHAT2
```

**Step 3**  Run the following commands to decompress the TopHat2 installation package:

```
tar xvf tophat-2.1.1.tar.gz
```

**Step 4**  Prepare a PATCH file and copy the following content to the `tophat.patch` file in the same directory as the `tophat-2.1.1` directory.

1. **vi tophat.patch**

2. **Press I to enter the editing mode and add the following content:**

```
diff -ubw -aur tophat-2.1.1/src/juncs_db.cpp tophat-2.1.1-patched/src/juncs_db.cpp
--- tophat-2.1.1/src/juncs_db.cpp 2016-02-14 19:21:17.354079000 +0100
+++ tophat-2.1.1-patched/src/juncs_db.cpp 2016-06-02 11:26:34.097425159 +0200
@@ -338,7 +338,7 @@
    uint32_t left_coord = atoi(scan_left_coord);
    uint32_t right_coord = atoi(scan_right_coord);
    bool antisense = *orientation == '-';
-   junctions.insert(make_pair<Junction, JunctionStats>(Junction(ref_id, left_coord, right_coord,
+   junctions.insert(make_pair<Junction, JunctionStats>(Junction(ref_id, left_coord, right_coord, antisense),
                                                 JunctionStats()));
   +   junctions.insert(make_pair(Junction(ref_id, left_coord, right_coord, antisense),
                                   JunctionStats()));
 }
```

```
diff -ubw -aur tophat-2.1.1/src/tophat_reports.cpp tophat-2.1.1-patched/src/tophat_reports.cpp
--- tophat-2.1.1/src/tophat_reports.cpp 2016-02-23 22:20:44.320710000 +0100
+++ tophat-2.1.1-patched/src/tophat_reports.cpp 2016-06-02 11:26:22.057239478 +0200
@@ -2705,7 +2705,7 @@
     -   gtf_junctions.insert(make_pair<Junction, JunctionStats>(Junction(ref_id, left Coord, right Coord, antisense), junction_stat));
-   +   gtf_junctions.insert(make_pair<Junction, JunctionStats>(Junction(ref_id, left Coord, right Coord, antisense),
```

---End
junction_stat);
    }
    fprintf(stderr, "Loaded %d GFF junctions from %s.\n", (int)(gtf_junctions.size()),
    gtf_juncs.c_str());

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

**Step 5** Run the following command to install the PATCH file:

```
patch -Np0 -i tophat.patch
```

**Step 6** Run the following command to switch to the directory generated after the package is decompressed:

```
cd tophat-2.1.1
```

**Step 7** Run the following command perform configuration:

```
./configure --prefix=/path/to/TOPHAT2 --with-boost=/path/to/BOOST
```

**Step 8** Run the following commands to perform compilation and installation:

```
make -j 1
make install
```

----End

### 4.15.7 Running and Verifying TopHat2

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to set TopHat2 environment variables:

```
export PATH=/path/to/TOPHAT2/bin:$PATH
```

**Step 3** Run the following command to switch to the directory where the test case is stored:

```
cd /path/to/CASE
```

**Step 4** Run the following command to decompress the test case:

```
tar xvf test_data.tar.gz
```

**Step 5** Run the following command to switch to the directory where the test case is stored:

```
cd test_data
```

**Step 6** Run the following command to run the TopHat2 test:

```
tophat2 -r 20 test_ref reads_1.fq reads_2.fq
```

If Run complete is displayed after the case is executed, the case is running properly. If FAILED is displayed, a running error occurs.

View the real time value in the last time command output in the log. The unit is \( x \) seconds. The lower the value is, the better the performance is.
4.15.8 More Information

Patch file:  
https://aur.archlinux.org/cgit/aur.git/tree/make_pair.patch?h=tophat

4.16 Bowtie2 2.4.1 Porting Guide (CentOS 7.6)

4.16.1 Introduction

Bowtie 2 is a tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100 or 1000 of characters, and particularly good at aligning to relatively long (e.g. mammalian) genomes.

It is usually the first step in aligning the genomic (including identifying variation calling, ChIP-seq, RNA-seq, BS-seq) pipelines.

It can handle very long readings (for example, 10s or 100s of kilobytes), but it optimizes the length and error pattern of readings generated by recent sequencers, such as Illumina HiSeq 2000, Roche 454, and Ion Torrent instruments.

Bowtie 2 indexes the genome with an FM Index (based on Burrows-Wheeler Transform or BWT) to keep its memory footprint small. For the human genome,
the memory footprint is about 3.2 GB. Bowtie 2 supports gapped, local, and end-to-end alignment modes. Multiple processors can be used at the same time to greatly improve the alignment speed.

For details about AmberTools, visit the official Bowtie 2 website.

Programming language: C


Open-source license: GPL 3.0

**Recommended Version**

Bowtie 2.4.1

**4.16.2 Environment Requirements**

**Hardware Requirements**

*Table 4-81* lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

**Software Requirements**

*Table 4-82* lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>bowtie2</td>
<td>2.4.1</td>
<td><a href="https://jaist.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.4.1/bowtie2-2.4.1-source.zip">https://jaist.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.4.1/bowtie2-2.4.1-source.zip</a></td>
</tr>
<tr>
<td>Python3</td>
<td>3.7.0</td>
<td><a href="https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz">https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz</a></td>
</tr>
<tr>
<td>SIMDe</td>
<td>master</td>
<td><a href="https://github.com/nemequ/simde">https://github.com/nemequ/simde</a></td>
</tr>
<tr>
<td>sse2neno</td>
<td>master</td>
<td><a href="https://github.com/jratcliff63367/sse2neon/archive/master.zip">https://github.com/jratcliff63367/sse2neon/archive/master.zip</a></td>
</tr>
<tr>
<td>Test case</td>
<td>chromFa.tar.gz</td>
<td><a href="http://hgdownload.cse.ucsc.edu/goldenPath/mm10/bigZips/chromFa.tar.gz">http://hgdownload.cse.ucsc.edu/goldenPath/mm10/bigZips/chromFa.tar.gz</a></td>
</tr>
</tbody>
</table>
OS Requirements

Table 4-83 lists the OS requirements.

Table 4-83 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.16.3 Planning the Paths for Software Porting

Table Paths for software porting lists the software installation paths involved in the Bowtie 2 software porting.

Table 4-84 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see <a href="#">Planning the Installation Paths</a> in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/BOWTIE2</td>
<td>Installation path of Bowtie2</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/PYTHON3</td>
<td>Installation path of Python 3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/SIMDE</td>
<td>Installation path of SIMDe</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/SSE2NEON</td>
<td>Installation path of SSE2NEON</td>
<td></td>
</tr>
</tbody>
</table>

4.16.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.
**Procedure**

**Table 4-85 Configuration process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see section &quot;Setting Up the Environment for the Cluster Scenario&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Install Python 3.</td>
<td>For details, see 4.16.4.1 Installing Python 3.</td>
</tr>
</tbody>
</table>

### 4.16.4.1 Installing Python 3

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to decompress the Python 3 installation package:

```
tar -xvf Python-3.7.0.tgz
```

**Step 3** Run the following command to switch to the directory generated after the package is decompressed:

```
cd Python-3.7.0
```

**Step 4** Run the following commands to compile and install Python 3:

```
./configure --prefix=/path/to/PYTHON3
make -j16
make install
```

**Step 5** Run the following command to set the Python3 environment variable:

```
export PATH=/path/to/PYTHON/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON/lib:$LD_LIBRARY_PATH
```

**Step 6** Verify whether the Python 3 environment has taken effect.

```
python -V
```

### 4.16.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the Bowtie 2 installation package `bowtie2-2.4.1-source.zip`.

Download address: [https://jaist.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.4.1/bowtie2-2.4.1-source.zip](https://jaist.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.4.1/bowtie2-2.4.1-source.zip).
Step 2 Use SFTP to upload the Bowtie 2 installation package to the /path/to/BOWTIE2 directory on the server.

----End

4.16.6 Compiling and Installing bowtie 2

4.16.6.1 Installing Bowtie 2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install zlib.aarch64 zlib-devel.aarch64 zlib-static.aarch64 using YUM:

```
yum install zlib.aarch64 zlib-devel.aarch64 zlib-static.aarch64
```

Step 3 Run the following command to decompress the installation package:

```
unzip bowtie2-2.4.1-source.zip
```

Step 4 Run the following command to switch to the directory generated after the package is decompressed:

```
cd bowtie2-2.4.1
mkdir -p /third_party/simde/simde/x86/
```

Step 5 Run the following command to perform configuration:

```
export NO_TBB=1
export POPCNT_CAPABILITY=0
```

Step 6 Run the following command to decompress SIMDe:

```
unzip simde-master.zip
```

Step 7 Copy the SIMDe file to BOWTIE2.

```
cp -a /path/to/SIMDE/simde-master/simde /path/to/BOWTIE2/bowtie2-2.4.1/third_party/simde/
```

Step 8 Run the following commands to decompress the package:

```
unzip master.zip
```

Step 9 Run the following command to switch to the directory generated after the package is decompressed:

```
cd sse2neon-master
```

Step 10 Copy the sse2neon.h file to BOWTIE2.

```
cp -a SSE2NEON.h /path/to/BOWTIE2/ bowtie2-2.4.1/third_party/simde/simde/x86/
```

Step 11 Run the following command to complete compilation:
cd .. /bowtie2-2.4.1
make all

Step 12 Run the following commands to set environment variables:
export PATH=/path/to/BOWTIE2/bowtie2-2.4.1:SPATH

----End

4.16.6.2 Creating bowtie 2 Index

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following command to switch to the directory generated after the package is decompressed:
tar -zxvf chromFa.tar.gz
Step 3 Run the following command to complete redirection:
cat *.fa > mm10.fa
Step 4 Run the following command to create an index:
bowtie2-build mm10.fa mm10

----End

4.16.7 Running and Verifying bowtie 2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following commands to run Bowtie 2 and obtain the SAM file:
bowtie2 -p 6 -3 5 --local -x mm10 -1 bowtie2-2.4.1/example/reads/reads_1.fq -2 bowtie2-2.4.1/example/reads/reads_2.fq -S example.sam

The following is an example of the test result.

<table>
<thead>
<tr>
<th>Reads</th>
<th>Count</th>
<th>Alignment Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>10000 (100.00%)</td>
<td>10000 (100.00%) aligned concordantly 0 times</td>
</tr>
<tr>
<td></td>
<td>0 (0.00%)</td>
<td>0 (0.00%) aligned concordantly &gt;1 times</td>
</tr>
<tr>
<td></td>
<td>0.00% overall alignment rate</td>
<td></td>
</tr>
</tbody>
</table>

----End

4.16.8 More Information

Bowtie2 Installation Guide (in Chinese only):
https://blog.csdn.net/u011262253/article/details/79833969

Download chromFa.tar.gz.
4.17 FastQC 0.11.9 Porting Guide (CentOS 7.6)

4.17.1 Introduction

FastQC is a quality control tool for high-throughput sequence data. It aims to provide a simple method to check the quality of original sequence data from high-throughput sequencing pipes.

For more information, visit the official FastQC website.

Programming language: C/Fortran 90

Brief description: a quality control tool for gene sequence data.

Open-source license: GPL 3.0

Recommended Version

fastqc_v0.11.9

4.17.2 Environment Requirements

Hardware Requirements

Table 4-86 lists the hardware requirements.

Table 4-86 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-87 lists the software requirements.

Table 4-87 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>FastQC</td>
<td>0.11.9</td>
<td><a href="http://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.9.zip">http://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.9.zip</a></td>
</tr>
<tr>
<td>openJDK</td>
<td>1.8.0</td>
<td>Provided by the system image</td>
</tr>
<tr>
<td>Test case</td>
<td>B17NC_R1</td>
<td><a href="https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz">https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz</a></td>
</tr>
</tbody>
</table>
OS Requirements

Table 4-88 lists the OS requirements.

Table 4-88 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.17.3 Planning the Paths for Software Porting

Table 4-89 lists the software installation paths involved in the FastQC software porting.

Table 4-89 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/FASTQC</td>
<td>Installation path of FastQC</td>
<td>The installation path listed in this table is only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

4.17.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.
## Configuration Process

### Table 4-90 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the <strong>HPC Solution Basic Environment Setup Guide</strong>.</td>
</tr>
<tr>
<td>2</td>
<td>Install JDK 1.8.</td>
<td>For details, see <strong>4.17.4.1 Installing JDK 1.8.</strong></td>
</tr>
<tr>
<td>3</td>
<td>Install the GUI.</td>
<td>For details, see <strong>4.17.4.2 Installing the GUI.</strong></td>
</tr>
</tbody>
</table>

### 4.17.4.1 Installing JDK 1.8.

**Procedure**

**Step 1** Use PuTTY to log in to the server as the *root* user.

**Step 2** Check whether OpenJDK is installed. If yes, skip **Step 3**.

```java
java -version
```

**Note**

FastQC requires that the OpenJDK version be later than 1.6.0. Otherwise, compatibility issues may occur.

**Step 3** Run the following command to install JDK1.8:

```bash
yum install java* -y
```

--- End

### 4.17.4.2 Installing the GUI

**Procedure**

**Step 1** Run the following command to install the X11 GUI:

```bash
yum -y groupinstall "Server with GUI"
```

--- End

### 4.17.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the FastQC installation package **fastqc_v0.11.9.zip**.
Step 2 Use SFTP to upload the FastQC installation package to the `/path/to/FastQC` directory on the server.

----End

4.17.6 Compiling and Installing FastQC

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to create a main program installation directory:

```bash
mkdir -p /path/to/FASTQC
```

Step 3 Run the following command to copy the installation package to the main program installation directory:

```bash
cp fastqc_v0.11.9.zip /path/to/FASTQC
```

Step 4 Run the following command to go to the main program installation directory:

```bash
cd /path/to/FASTQC
```

Step 5 Run the following commands to decompress the package:

```bash
unzip fastqc_v0.11.9.zip
```

Step 6 Run the following command to switch to the directory generated after the package is decompressed:

```bash
cd FastQC
```

Step 7 Run the following command to configure the address displayed on the GUI:

```bash
export DISPLAY=IP address of the server that can be connected to the PC:0.0
```

Step 8 Run the following command to verify that the fastqc script is available:

```bash
chmod +755 fastqc
./fastqc
```

The following is an example of the test result.
4.17.7 Running and Verifying FastQC

Procedure (Using CLI)

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to create a test directory:

```bash
mkdir -p /path/to/FASTQC/test
```

Step 3 Run the following commands to copy the test case to the test directory and switch to the test directory:

```bash
cp B17NC_R1.fastq.gz /path/to/FASTQC/test
cd /path/to/FASTQC/test
```

Step 4 Run the following command to decompress the test case:

```bash
gzip -d B17NC_R1.fastq.gz
```

Step 5 Run the following command to run the test:

```bash
/path/to/FASTQC/FastQC/fastqc -o ./ /B17NC_R1.fastq
```

Step 6 After the execution is complete, the B17NC_R1_fastqc.html and B17NC_R1_fastqc.zip files are generated. Run the following command to check whether the files exist:

```bash
ll
```

Use a browser to open the HTML file and view the result.

---End
Procedure (Using GUI)

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to log in to the GUI:

`/path/to/FASTQC/FastQC/fastqc`

**Step 3** Click **File > Open** in the upper left corner and select a case.

**Step 4** Select a test case and click **Open**. If the following page is displayed, the test is running.
Figure 4-12 shows an example of the test result.
4.17.8 More Information

Official FastQC website:

http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

FastQC installation guide:

http://www.bioinformatics.babraham.ac.uk/projects/fastqc/INSTALL.txt

4.18 Beagle 5.1 Porting Guide (CentOS 7.6)

4.18.1 Introduction

Beagle is one of the common software for genotype imputation. The latest version is 5.1, which greatly improves the accuracy and running speed. Compared with other software of the same type, Beagle has two obvious advantages: low memory consumption and fast running speed.

For more information about Beagle, visit the official Beagle website.

Language: Java

One-sentence description: Genotype imputation software.

Open-source license: GPL 3.0

Recommended Software Version

Beagle 5.1

4.18.2 Environment Requirements

Hardware Requirements

Table 4-91 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-92 lists the software requirements.
### Table 4-92 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beagle</td>
<td>5.1</td>
<td><a href="http://faculty.washington.edu/browning/beagle/beagle.27Apr20.b81.jar">http://faculty.washington.edu/browning/beagle/beagle.27Apr20.b81.jar</a></td>
<td>Beagle tool package</td>
</tr>
<tr>
<td>bref3.2 7Apr20.b81.jar</td>
<td>-</td>
<td><a href="http://faculty.washington.edu/browning/beagle/bref3.27Apr20.b81.jar">http://faculty.washington.edu/browning/beagle/bref3.27Apr20.b81.jar</a></td>
<td>Tool package for converting the VCF format to the Bref3 format</td>
</tr>
<tr>
<td>unbref3 27Apr20.b81.jar</td>
<td>-</td>
<td><a href="http://faculty.washington.edu/browning/beagle/unbref3.27Apr20.b81.jar">http://faculty.washington.edu/browning/beagle/unbref3.27Apr20.b81.jar</a></td>
<td>Tool package for converting the Bref3 format to the VCF format</td>
</tr>
<tr>
<td>1000 Genomes Project data</td>
<td>-</td>
<td><a href="http://faculty.washington.edu/browning/beagle/test.27Apr20.b81.vcf.gz">http://faculty.washington.edu/browning/beagle/test.27Apr20.b81.vcf.gz</a></td>
<td>Test verification file</td>
</tr>
</tbody>
</table>

### OS Requirements

**Table 4-93** lists the OS requirements.

### Table 4-93 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.18.3 Planning the Paths for Software Porting

**Table 4-94** lists the software installation paths involved in the Beagle software porting.
Table 4-94 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning Data for Installation in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>4</td>
<td>/path/to/BEAGLE</td>
<td>Installation path of Beagle</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

4.18.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration process

Table 4-95 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Setting up the Java environment</td>
<td>For details, see 4.18.4.1 Setting Up the Java Environment.</td>
</tr>
</tbody>
</table>

4.18.4.1 Setting Up the Java Environment

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to install the Java package:

```bash
yum install java-1.8.0-openjdk.aarch64
```
Step 3 Run the following command to check whether Java has been installed:

```
java -version
```

If the following information is displayed, Java has been installed successfully:
```
openjdk version "1.8.0_212"
```

----End

4.18.5 Obtaining the Source Code

Procedure

Step 1 Download the Beagle package `beagle.27Apr20.b81.jar`.

Download address: [http://faculty.washington.edu/browning/beagle/beagle.27Apr20.b81.jar](http://faculty.washington.edu/browning/beagle/beagle.27Apr20.b81.jar).

Step 2 Download the tool package `bref3.27Apr20.b81.jar` for converting VCF files to Bref3 files.


Step 3 Download the tool package `unbref3.27Apr20.b81.jar` for converting Bref3 files to VCF files.


Step 4 Download the test file `test.27Apr20.b81.vcf.gz`.

Download address: [http://faculty.washington.edu/browning/beagle/test.27Apr20.b81.vcf.gz](http://faculty.washington.edu/browning/beagle/test.27Apr20.b81.vcf.gz).

Step 5 Use SFTP to upload the downloaded installation packages to the `/path/to/Beagle` directory on the server.

----End

4.18.6 Running and Verifying Beagle

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to switch to the Beagle folder:

```
cd /path/to/BEAGLE
```

Step 3 Run the following commands to create and edit the file:

1. Run the vi command to open the `test.sh` file.

```
vim test.sh
```

2. Press I to enter the editing mode and add the following content to the script:

```
#!/bin/sh

echo "*** Creating test files: ref.27Apr20.b81.vcf.gz target.27Apr20.b81.vcf.gz ***"
```

----End
```bash
echo zcat test.27Apr20.b81.vcf.gz | cut -f1-190 | tr '/' '|' | gzip > ref.27Apr20.b81.vcf.gz
echo zcat test.27Apr20.b81.vcf.gz | cut -f1-9,191-200 | gzip > target.27Apr20.b81.vcf.gz

echo *** Running test analysis with "gt=" argument ***
java -jar beagle.27Apr20.b81.jar gt=test.27Apr20.b81.vcf.gz out=out.gt

echo *** Running test analysis with "ref=" and "gt=" arguments ***
java -jar beagle.27Apr20.b81.jar ref=ref.27Apr20.b81.vcf.gz gt=target.27Apr20.b81.vcf.gz out=out.ref

echo *** Making "bref3" file ***
java -jar bref3.27Apr20.b81.jar ref.27Apr20.b81.vcf.gz > ref.27Apr20.b81.bref3

echo *** Running test analysis with "bref3" file ***
java -jar beagle.27Apr20.b81.jar ref=ref.27Apr20.b81.bref3 gt=target.27Apr20.b81.vcf.gz out=out.bref3
```

**NOTE**

```
java -Xmx [ GB ] g -jar beagle.jar [ arguments ]
```

- In the preceding command, [GB] indicates the upper limit of the memory pool, and [arguments] indicates the arguments separated by spaces.
- Two arguments must be added: `gt = [file]` specifies a .vcf file that contains the genotype data of the research sample, and `out = [string]` specifies the prefix of the output file name.
- For details about more parameters, see the official document at [http://faculty.washington.edu/browning/beagle/beagle_5.1_08Nov19.pdf](http://faculty.washington.edu/browning/beagle/beagle_5.1_08Nov19.pdf).

3. Press `Esc`, enter `.wq!`, and press `Enter` to save the file and exit.

**Step 4** Run the following command to grant the execute permission on the script:

```bash
chmod a+x test.sh
```

**Step 5** Run the following command to execute the script:

```bash
./test.sh
```

The following is an example of the command output. The command output is also saved to the .log file.

**Cumulative Statistics:**

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study markers</td>
<td>1,356</td>
</tr>
<tr>
<td>Haplotype phasing time</td>
<td>1 second</td>
</tr>
<tr>
<td>Total time</td>
<td>1 second</td>
</tr>
<tr>
<td>End time</td>
<td>10:33 AM CST on 09 May 2020</td>
</tr>
</tbody>
</table>

beagle.27Apr20.b81.jar finished

4.18.7 More Resources

Official Beagle website:

[https://faculty.washington.edu/browning/beagle/beagle.html](https://faculty.washington.edu/browning/beagle/beagle.html)
4.19 Bowtie 1.2.3 Porting Guide (CentOS 7.6)

4.19.1 Introduction

Bowtie is an ultrafast, memory-efficient short read aligner geared toward quickly aligning large sets of short DNA sequences (reads) to large genomes.

Bowtie indexes the genome with a Burrows-Wheeler index to keep its memory footprint small: for the human genome, the index is typically about 2.2 GB (for unpaired alignment) or 2.9 GB (for paired-end or colorspace alignment). Multiple processors can be used simultaneously to achieve greater alignment speed.

For more information about Bowtie, visit the official Bowtie website.

Programming language: C

Brief description: ultrafast, memory-efficient short read aligner, used in the next generation of sequencers for short DNA sequences

Open-source license: GPL 3.0

Recommended Version

Bowtie 1.2.3

4.19.2 Environment Requirements

Hardware Requirements

Table 4-96 lists the hardware requirements.

Table 4-96 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-97 lists the software requirements.

Table 4-97 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie</td>
<td>1.2.3</td>
<td><a href="https://codeload.github.com/BenLangmead/bowtie/zip/v1.2.3">https://codeload.github.com/BenLangmead/bowtie/zip/v1.2.3</a></td>
</tr>
<tr>
<td>simde</td>
<td>master</td>
<td><a href="https://codeload.github.com/nemequ/simde/zip/master">https://codeload.github.com/nemequ/simde/zip/master</a></td>
</tr>
</tbody>
</table>
### OS Requirements

Table 4-98 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.19.3 Planning the Paths for Software Porting

Table Paths for software porting lists the software installation paths involved in the Bowtie software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see section Planning Data for Installation in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/BOWTIE</td>
<td>Installation path of Bowtie</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/SIMDE</td>
<td>Installation path of SIMDE</td>
<td></td>
</tr>
</tbody>
</table>

Issue 14 (2021-07-23)  Copyright © Huawei Technologies Co., Ltd.  542
### 4.19.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Cluster Environment&quot; in <em>HPC Solution Basic Environment Setup Guide.</em></td>
</tr>
</tbody>
</table>

#### 4.19.5 Obtaining the Source Code

**Procedure**

**Step 1**  Download the following software packages and case files.
- Download the Bowtie installation package *bowtie-1.2.3.zip.*
  URL: [https://github.com/BenLangmead/bowtie/archive/v1.2.3.zip](https://github.com/BenLangmead/bowtie/archive/v1.2.3.zip)
- Download the simde-master installation package *simde-master.zip.*
  URL: [https://codeload.github.com/nemequ/simde/zip/master](https://codeload.github.com/nemequ/simde/zip/master)
- Download the sse2neon-master installation package *sse2neon-master.zip.*
  URL: [https://codeload.github.com/jratcliff63367/sse2neon/zip/master](https://codeload.github.com/jratcliff63367/sse2neon/zip/master)
- Download the human genome package *chromFa.tar.gz.*
  URL: [https://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz](https://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz)
- Download the *SRR742200_1.fastq* test case file *SRR742200_1.fastq.gz* from:
Download the SRR742200_2.fastq test case file SRR742200_2.fastq.gz from:
ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR742/SRR742200/SRR742200_2.fastq.gz

Step 2 Upload the software packages and case files to the server using an SFTP tool.

- Upload the Bowtie installation package to the /path/to/BOWTIE directory on the server.
- Upload the simde-master installation package to the /path/to/SIMDE directory on the server.
- Upload the sse2neon-master installation package to the /path/to/SSE2NENO directory on the server.
- Upload the chromFa.tar.gz installation package to the /path/to/CHROMFA directory on the server.
- Upload the SRR742200_1.fastq.gz file to the /path/to/CHROMFA/man_hg19/ directory on the server.
- Upload the SRR742200_2.fastq.gz file to the /path/to/CHROMFA/man_hg19/ directory on the server.

----End

4.19.6 Compiling and Installing Bowtie

4.19.6.1 Installing Bowtie

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies:

```
yum install zlib.aarch64 zlib-devel.aarch64 zlib-static.aarch64
```

Step 3 Run the following command to decompress the installation package:

```
unzip bowtie-1.2.3.zip
```

Step 4 Run the following command to switch to the directory that contains decompressed files:

```
cd /path/to/BOWTIE/bowtie-1.2.3
```

Step 5 Run the following commands to modify the configuration file Makefile:

1. Run the vi command to open the Makefile file.

```
vi Makefile
```

2. Press i to go to the edit mode.

To support aarch64, add the following information in bold to the end of about line 162:

```
BIT5=32
ifeq (x86_64,$(shell uname -m))
    BITS=64
endif
ifeq (aarch64,$(shell uname -m))
    BITS=32
endif
```

```
Modify the following content in about line 187:

```
DEBUG_FLAGS = -O0 -g3
RELEASE_FLAGS = -O3
NOASSERT_FLAGS = -DNDEBUG
FILE_FLAGS = -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE
```

Add the following content in bold after the commented lines to add the aarch64 branch:

```
M64_FLAGS := -m64
ifeq (aarch64,$(shell uname -m))
M64_FLAG :=
endif
```

3. Press Esc, type :wq!, and press Enter to save the file and exit.

**Step 6** Run the following commands to modify the alphabet.cpp file:

1. Run the vi command to open the alphabet.cpp file.
   
   ```
   vi /path/to/BOWTIE/bowtie-1.2.3/alphabet.cpp
   ```

2. Press i to go to the edit mode and modify about line 276:

   **Before the modification:**
   ```
   char mask2iupac[16] = {...}
   ```

   **After the modification:**
   ```
   signed char mask2iupac[16] = {...}
   ```

3. Press Esc, type :wq!, and press Enter to save the file and exit.

**Step 7** Run the following command to create a folder:

```
mkdir -p ./third_party/simde/simde/x86
```

**Step 8** Run the following command to perform the configuration:

```
export NO_TBB=1
export POPCNT_CAPABILITY=0
```

**Step 9** Run the following command to decompress the SIMDE installation package:

```
unzip simde-master.zip
```

**Step 10** Copy the simde directory and its contents in simde-master to bowtie-1.2.3/third_party/simde/.

```
cp -a simde-master/simde /path/to/BOWTIE/bowtie-1.2.3/third_party/simde/
```

**Step 11** Run the following command to decompress the SSE2NEON installation package:

```
unzip sse2neon-master.zip
```

**Step 12** Copy the SSE2NEON.h file to bowtie-1.2.3/third_party/simde/simde/x86/.

```
cp -a sse2neon-master/SSE2NEON.h /path/to/BOWTIE/bowtie-1.2.3/third_party/simde/simde/x86/
```

**Step 13** Run the following command to compile Bowtie:

```
cd /path/to/BOWTIE/bowtie-1.2.3
make
```
Step 14  Run the following command to perform installation:

```
make install
```

Step 15  Run the following commands to set the environment variable:

```
export PATH=/path/to/BOWTIE/bowtie-1.2.3:$PATH
```

----End

4.19.6.2 Creating Bowtie Index

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to switch to the chromFa directory:

```
 cd /path/to/CHROMFA
```

Step 3  Run the following command to decompress the chromFa installation package.

```
 tar -xzvf chromFa.tar.gz
```

Step 4  Run the following command to complete redirection:

```
 cat *.fa > man.fa
```

Step 5  Run the following command to create an index:

```
 bowtie-build --threads 128 man.fa man_index
```

Step 6  Create the man_hg19 directory and place the man_index*.ebwt file produced in the above step to the directory.

```
 mkdir man_hg19
 mv *.ebwt man_hg19
```

----End

4.19.7 Running and Verifying Bowtie

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to switch to the man_hg19 directory:

```
 cd /path/to/CHROMFA/man_hg19/
```

Step 3  Run the following commands to decompress the case files:

```
 gzip -d SRR742200_1.fastq.gz
 gzip -d SRR742200_2.fastq.gz
```

Step 4  Run the following command to run Bowtie to obtain the .sam file (the corresponding biological sequence text xxx.fastq is required):

```
bowtie -S -t -q -p 128 man_index -1 ./SRR742200_1.fastq -2 ./SRR742200_2.fastq SRR742200_out.sam

The following is an example of the result:

----End

4.19.8 Troubleshooting

Failure 1: An Error Occurs During the Bowtie Installation

Symptom

When running the make command to perform compilation during Bowtie installation, an error occurs. The error message is "narrowing conversion of '-1' from 'int' to 'char' inside { } [-Wnarrowing]".

Possible Causes

There is a narrowing conversion error.

Procedure

Step 1 Run the following commands to modify the alphabet.cpp file:

1. vi /path/to/BOWTIE/bowtie-1.2.3/alphabet.cpp
2. Press i to enter the insert mode and modify about line 276.

   Before the modification:
   char mask2iupac[16] = {...}

   After the modification:
   signed char mask2iupac[16] = {...}

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

----End

4.19.9 More Information

Bowtie installation guide:

https://bbs.huaweicloud.com/forum/thread-47080-1-1.html?

Download chromFa.tar.gz:

https://hgdownload.soe.ucsc.edu/downloads.html#mm9pairwise-primate

4.20 pblat 2.1 Porting Guide (CentOS 7.6)

4.20.1 Introduction

For DNA sequences, BLAST-like alignment tool (blat) is used to design and find sequences of 95% and greater similarity of length 40 bases or more. For protein
sequences, blat is used to design and find sequences of 80% and greater similarity of length 20 amino acids or more. pblat is a multi-threaded blat algorithm that fully utilizes the advantages of multi-core processors and greatly reduces the running time with the same memory consumption.

For more information about pblat, visit the official pblat website.

Language: C

One-sentence description: A BLAST-like alignment tool that supports multiple threads.

Recommended Software Version

pblat 2.1

4.20.2 Environment Requirements

Hardware Requirements

*Table 4-101* lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

*Table 4-102* lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>pblat</td>
<td>2.1</td>
<td><a href="https://codeload.github.com/icebert/pblat/tar.gz/2.1">https://codeload.github.com/icebert/pblat/tar.gz/2.1</a></td>
</tr>
<tr>
<td>NONCODEv5_human.fa</td>
<td></td>
<td><a href="http://www.noncode.org/datadownload/NONCODEv5_human.fa.gz">http://www.noncode.org/datadownload/NONCODEv5_human.fa.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

*Table 4-103* lists the OS requirements.
4.20.3 Planning the Paths for Software Porting

Table 4-104 lists the software installation paths involved in the pblat software porting.

Table 4-104 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning Data for Installation in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/PBLAT</td>
<td>Installation path of pblat</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

4.20.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 4-105 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see Setting Up the Cluster Environment in HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>
4.20.5 Obtaining the Source Code

Procedure

Step 1  Download the pblat installation package pblat-2.1.tar.gz.
Download address: https://codeload.github.com/icebert/pblat/tar.gz/2.1.

Step 2  Use SFTP to upload the pblat installation package to the /path/to/PBLAT directory on the server.

----End

4.20.6 Compiling and Installing pblat

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to decompress the pblat installation package:

tar -zxvf pblat-2.1.tar.gz

Step 3  Run the following command to switch to the directory that contains decompressed files:

cd pblat-2.1/

Step 4  Run the following commands to modify the Makefile file.
1. Run the vim command to open the Makefile file.
   
vim Makefile
2. Press I to enter the edit mode and modify the following information:
   
   MACHTYPE=aarch64
   CC=gcc
3. Press Esc, type :wq!, and press Enter to save the file and exit.

Step 5  Run the following commands to modify the htmshell.c file:
1. Run the vim command to open the htmshell.c file.
   
vim lib/htmshell.c
2. Run the following command to locate line 714:
   
   :714
3. Press I to enter the edit mode and modify the content in line 714:
   
   if (format != NULL)
4. Press Esc, type :wq!, and press Enter to save the file and exit.

Step 6  Run the following commands to compile pblat:

make -j16

Move the executable program pblat to the planned path.

cp pblat /path/to/PBLAT

Step 7  Run the following command to add environment variables:
export PATH=/path/to/PBLAT:$PATH

4.20.7 Running and Verifying pblat

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to create a compute instance folder and copy the compute instance file:

```
mkdir /path/to/PBLAT/test

cp hg19.fa /path/to/PBLAT/test

cp NONCODEv5_human.fa.gz /path/to/PBLAT/test
```

**Step 3** Run the following command to go to the compute instance directory:

```
cd /path/to/PBLAT/test
```

**Step 4** Run the following command to process the compute instance file:

```
gzip -d NONCODEv5_human.fa.gz

cat NONCODEv5_human.fa | grep -v "^#" | head -799996 > test-799996.fa
```

**Step 5** Run the following command to perform the test:

```
{ time pblat -threads=96 hg19.fa test-799996.fa out.psl ; } 2>&1 | tee pblat-kunpeng.log
```

You need to check the value of real in the pblat-kunpeng.log file. The unit is s. A smaller value indicates better performance.

*Figure 4-13* is an example of the output result.

*Figure 4-13 Result example*

> Searched 251927 bases in 260 sequences
> Searched 240170 bases in 260 sequences
> Searched 314326 bases in 260 sequences
> Searched 257271 bases in 260 sequences
> Searched 255877 bases in 260 sequences
> Searched 352678 bases in 260 sequences
> Searched 374791 bases in 260 sequences
> Searched 277221 bases in 260 sequences
> Searched 304065 bases in 260 sequences
> Searched 382588 bases in 260 sequences
> Searched 405694 bases in 260 sequences

 real 1539.22
user 42070.01
sys 10.29

----End
4.20.8 Troubleshooting

Problem 1: An Error Is Reported When the make Command Is Executed

Symptom
An error message " error: invalid operands to binary != (have'va_list '{aka' __va_list '}' and' void * ')" is displayed when the make command is executed.

Possible Cause
The source code programming language is not standard.

Procedure
Modify line 714 in the htmshell.c file by referring to Step 5.

End

4.20.9 More Resources

pblat installation and running guide:
https://github.com/icebert/pblat

4.21 SnpEff V4_3 Porting Guide (CentOS 7.6)

4.21.1 Introduction

SnpEff is a variant annotation and effect prediction tool. It annotates and predicts the effects of genetic variants (such as amino acid changes).

For more information about SnpEff, visit the official SnpEff website.

Language: Java
One-sentence description: A variant annotation and effect prediction tool.

Recommended Software Version

SnpEff V4_3.

4.21.2 Environment Requirements

Hardware Requirements

Table 4-106 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
Software Requirements

Table 4-107 lists the software requirements.

Table 4-107 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>SnpEff</td>
<td>V4.3</td>
<td><a href="https://excellmedia.dl.sourceforge.net/project/snpeff/snpEff_v4_3_core.zip">https://excellmedia.dl.sourceforge.net/project/snpeff/snpEff_v4_3_core.zip</a></td>
</tr>
<tr>
<td>GRCh 37.75</td>
<td>37.75</td>
<td><a href="https://nchc.dl.sourceforge.net/project/snpeff/databases/v4_3/snpEff_v4_3_GRCh37.75.zip">https://nchc.dl.sourceforge.net/project/snpeff/databases/v4_3/snpEff_v4_3_GRCh37.75.zip</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-108 lists the OS requirements.

Table 4-108 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.21.3 Planning the Paths for Software Porting

Table 4-109 lists the software installation paths involved in the SnpEff software porting.

Table 4-109 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning Data for Installation in HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>
4.21.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic</td>
<td>For details, see section Setting Up the Cluster Environment in HPC Solution</td>
</tr>
<tr>
<td></td>
<td>environment</td>
<td>Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

4.21.5 Obtaining the Source Code

Procedure

**Step 1** Download the SnpEff installation package `snpEff_v4_3_core.zip`.

Download address: [https://excellmedia.dl.sourceforge.net/project/snpeff/snpEff_v4_3_core.zip](https://excellmedia.dl.sourceforge.net/project/snpeff/snpEff_v4_3_core.zip)

**Step 2** Download the GRCh installation package `snpEff_v4_3_GRCh37.75.zip`.

Download address: [https://nchc.dl.sourceforge.net/project/snpeff/databases/v4_3/snpEff_v4_3_GRCh37.75.zip](https://nchc.dl.sourceforge.net/project/snpeff/databases/v4_3/snpEff_v4_3_GRCh37.75.zip)

**Step 3** Use SFTP to upload the SnpEff 37.75 installation package to the `/path/to/SNPEFF` directory on the server.

----End
4.21.6 Compiling and Installing Software

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to install the Java environment:

```
yum install java-1.8.0-openjdk-devel.aarch64 -y
```

Step 3 Run the following command to switch to the SnpEff directory:

```
cd /path/to/SNPEFF
```

Step 4 Run the following command to decompress the installation package:

```
unzip snpEff_v4_3_core.zip
```

Step 5 Run the following command to replicate the GRCh 37.75 file:

```
cp snpEff_v4_3_GRCh37.75.zip /path/to/SNPEFF/snpEff
```

Step 6 Run the following command to switch to the SnpEff directory:

```
cd /path/to/SNPEFF/snpEff
```

Step 7 Run the following command to decompress the installation package:

```
unzip snpEff_v4_3_GRCh37.75.zip
```

----End

4.21.7 Testing and Verification

Procedure

Step 1 Run the following command to list the databases (genomes) supported by SnpEff:

```
java -jar snpEff.jar databases
```

Step 2 Run the following command to switch to the SnpEff directory:

```
cd /path/to/SNPEFF/snpEff
```

Step 3 Run the following command to export the text with comments:

```
java -jar snpEff.jar -v GRCh37.75 /path/to/SNPEFF/snpEff/examples/test.chr22.vcf > test.chr22.out.vcf
```

After the execution is complete, three files are generated: snpEff_genes.txt, snpEff_summary.html, and test.chr22.out.vcf.

The following is an output example:
4.21.8 More Resources

SnpEff installation guide

http://snpeff.sourceforge.net/download.html#install

More databases:

https://sourceforge.net/projects/snpeff/files/databases/

4.22 Velvet 1.2.10 Porting Guide (CentOS 7.6)

4.22.1 Introduction

Velvet is a de novo genomic assembler specially designed for short read sequencing technologies, such as Solexa or 454, developed by Daniel Zerbino and Ewan Birney at the European Bioinformatics Institute (EMBL-EBI). Data in multiple libraries can be used at the same time.

For more information about Velvet, visit the official Velvet website.

Language: C

One-sentence description: A common genome assembly software, which is used to process second-generation sequencing data and has good assembly integrity.

Open-source license: GPL 3.0

Recommended Software Version

Velvet 1.2.10.
4.22.2 Environment Requirements

Hardware Requirements

Table 4-111 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-112 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velvet</td>
<td>1.2.10</td>
<td><a href="https://codeload.github.com/dzerbino/velvet/tar.gz/v1.2.10">https://codeload.github.com/dzerbino/velvet/tar.gz/v1.2.10</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-113 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.22.3 Planning the Paths for Software Porting

Table 4-114 lists the software installation paths involved in the Velvet software porting.
Table 4-114 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning Data for Installation in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/VELVET</td>
<td>Installation path of Velvet</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

4.22.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

Table 4-115 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section Setting Up the Cluster Environment in HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

4.22.5 Obtaining the Source Code

Procedure

Step 1  Download the Velvet installation package v1.2.10.tar.gz.

Download address: https://codeload.github.com/dzerbino/velvet.tar.gz/v1.2.10.

Step 2  Use SFTP to upload the Velvet installation package to the /path/to/Velvet directory on the server.

----End
4.22.6 Compiling and Installing Software

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to create a main program installation directory:

```
mkdir -p /path/to/VELVET
```

Step 3 Copy the installation package to the installation directory.

```
cp velvet_1.2.10.tgz /path/to/VELVET
```

Step 4 Run the following command to go to the main program installation directory:

```
cd /path/to/VELVET
```

Step 5 Run the following command to decompress the installation package:

```
tar -xvf velvet-1.2.10.tar.gz
```

Step 6 Run the following command to switch to the directory that contains decompressed files:

```
cd velvet-1.2.10
```

Step 7 Run the following commands to set the number of threads and the thread limit:

```
export OMP_NUM_THREADS=96
export OMP_THREAD_LIMIT=95
```

- **NOTE**
  - `OMP_NUM_THREADS` indicates the number of threads, which is equal to the number of CPU cores.
  - `OMP_THREAD_LIMIT` indicates the thread limit, which is equal to the number of CPU cores minus 1.

Step 8 Run the following command to modify the `Makefile` file:

```
sed -ri 's/\-m64/\-mabi=lp64/g' Makefile
```

Step 9 Run the following command to compile and install the software:

```
make 'GATECATEGORIES=10' 'MAXKMERLENGTH=57' 'LONGSEQUENCES=1' 'OPENMP=1' 'BUNDLEDZLIB=1'
```

After the compilation is complete, two executable files are generated: `velveth` and `velvetg`.

----End

4.22.7 Running and Verifying Velvet

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
**Step 2** Switch to the test directory.

```bash
cd /path/to/VELVET/velvet-1.2.10
```

**Step 3** Run the following command to run the executable file `velveth` to prepare data:

```bash
./velveth output/ 31 -shortPaired -fastq.gz tests/read1.fq.gz -shortPaired -fastq.gz tests/read2.fq.gz
```

**NOTE**
- **output**: indicates the output directory.
- **shortPaired**: specifies the sequencing type.
- **Fastaq**: specifies the sequence format. Different parameters are used to specify the format of the input file. `-fasta` corresponds to the fasta format. `-fastq` corresponds to the fastq format. `-fastq.gz` corresponds to the fastq.gz format. `-fasta.gz` corresponds to the fasta.gz format. `-sam`: corresponds to the sam format. `-ban` corresponds to the bam format.

The following is an example of the output result.

```
[0.000000] Reading FastQ file tests/read1.fq.gz;
[0.083994] 25000 sequences found
[0.084004] Done
[0.084266] Reading FastQ file tests/read2.fq.gz;
[0.162083] 25000 sequences found
[0.162091] Done
[0.364858] Reading read set file output//Sequences;
[0.376048] 50000 sequences found
[0.427913] Done
[0.427929] 50000 sequences in total.
[0.427937] Writing into roadmap file output//Roadmaps...
[1.182309]  === Sequences loaded in 0.713954 s
[1.303762] Done inputting sequences
[1.303770] Destroying splay table
[1.305914] Splay table destroyed
```

**Step 4** Run the following command to run the executable file `velvetg` to assemble the genome:

```bash
./velvetg output/ -min_contig_lgth 100
```

**NOTE**
- **output**: indicates the output directory.
- **min_contig_lgth**: indicates the minimum length of the Contig field. The Contig field whose length is less than the minimum length will be deleted and will not be displayed in the final result. After the running is complete, the `contigs.fai` file in the output directory is the final assembly result.

The following is an example of the output result.

```
[1.258599] Concatenation over!
[1.259034] Writing contigs into output//contigs.fa...
[1.292427] Writing into stats file output//stats.txt...
[1.327635] Writing into graph file output//LastGraph...
Final graph has 987 nodes and n50 of 199, max 2546, total 110172, using 0/50000 reads
```

----End
4.22.8 More Resources

View on GitHub:

https://github.com/dzerbino/velvet/tree/master

4.23 XHMM Porting Guide (CentOS 7.6)

4.23.1 Introduction

XHMM uses principal component analysis (PCA) normalization and a hidden Markov model (HMM) to detect and genotype copy number variation (CNV) from normalized read-depth data from targeted sequencing experiments. XHMM was explicitly designed to be used with targeted exome sequencing at high coverage (at least 60x - 100x) using Illumina HiSeq (or similar) sequencing of at least ~50 samples.

For more information about XHMM, visit the official XHMM website.

Language: C++

One-sentence description: tool for predicting copy number variation (CNV).

Open-source license: GPL v3

4.23.2 Environment Requirements

Hardware Requirements

Table 4-116 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-117 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>XHMM</td>
<td>Latest XHMM development version</td>
<td><a href="https://bitbucket.org/statgen/xhmm/get/master.zip">https://bitbucket.org/statgen/xhmm/get/master.zip</a></td>
</tr>
<tr>
<td>lapack</td>
<td>lapack v3.8.0</td>
<td><a href="https://github.com/Reference-LAPACK/lapack/archive/v3.8.0.tar.gz">https://github.com/Reference-LAPACK/lapack/archive/v3.8.0.tar.gz</a></td>
</tr>
</tbody>
</table>
### OS Requirements

Table 4-118 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.23.3 Planning the Paths for Software Porting

Table 4-119 lists the software installation paths involved in the XHMM software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning Data for Installation in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/LAPACK</td>
<td>Installation path of LAPACK</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/XHMM</td>
<td>Installation path of XHMM</td>
<td></td>
</tr>
</tbody>
</table>
### 4.23.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.

#### Configuration Process

Table 4-120 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section <strong>Setting Up the Cluster Environment</strong> in <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Installing OpneBLAS</td>
<td>For details, see <strong>4.23.4.1 Installing OpenBLAS</strong>.</td>
</tr>
<tr>
<td>3</td>
<td>Installing LAPACK</td>
<td>For details, see <strong>4.23.4.2 Installing lapack</strong>.</td>
</tr>
</tbody>
</table>

#### 4.23.4.1 Installing OpenBLAS

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the installation package:

\[
\text{tar xzvf OpenBLAS-0.3.9.tar.gz}
\]

**Step 3** Run the following command to switch to the directory that contains decompressed files:

\[
\text{cd OpenBLAS-0.3.9/}
\]

**Step 4** Run the following commands to set environment variables:

\[
\begin{align*}
\text{export CC} &= \text{`which gcc`} \\
\text{export CXX} &= \text{`which g++`} \\
\text{export FC} &= \text{`which gfortran`} \\
\end{align*}
\]
Step 5  Run the following command to compile and install OpenBLAS:

    make -j 96
    make PREFIX=/path/to/OPENBLAS install

Step 6  Run the following commands to set the OpenBLAS environment variables:

    export LIBRARY_PATH=/path/to/OPENBLAS:/lib:
    export LD_LIBRARY_PATH=/path/to/OPENBLAS:/lib:

----End

4.23.4.2 Installing lapack

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to decompress the installation package:

    tar xzvf lapack-3.8.0.tar.gz

Step 3  Run the following command to switch to the directory that contains decompressed files:

    cd lapack-3.8.0/

Step 4  Run the following command to configure the compiling file:

    cp make.inc.example make.inc

Step 5  Run the following command to compile and install LAPACK:

    make lapacklib -j96
    mkdir /path/to/LAPACK

Step 6  Run the following commands to set the LAPACK environment variables:

    export LIBRARY_PATH=/path/to/LAPACK:

----End

4.23.5 Obtaining the Source Code

Procedure

Step 1  Download the following software packages:

- Download the XHMM source code package statgen-xhmm-998f7c405974.zip.
  Download address: https://bitbucket.org/statgen/xhmm/get/master.zip
- Download the LAPACK source code package lapack-3.8.0.tar.gz.
  Download address: https://github.com/Reference-LAPACK/lapack/archive/v3.8.0.tar.gz
- Download the OpenBLAS source code package OpenBLAS-0.3.9.tar.gz.
4.23.6 Compiling and Installing XHMM

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the installation package:

```
unzip statgen-xhmm-*.zip
```

Step 3 Run the following command to switch to the directory that contains decompressed files:

```
cd statgen-xhmm-`
```

Step 4 Run the following command to edit the Makefile file:

1. Run the `vim` command to open the Makefile file.

```
vim Makefile
```

2. Press i to enter the edit mode and modify the content in line 89:

Before the change:
```
LAPACK_LIBS=lapack
```

After the change:
```
LAPACK_LIBS=openblas lapack
```

3. Press Esc, type :wq!, and press Enter to save the file and exit.

Step 5 Run the following commands to perform compilation and installation:

```
made -j 96
```

```
mkdir /path/to/XHMM
```

```
-p build/execs/xhmm params.txt /path/to/XHMM
```

Step 6 Run the following command to add the XHMM installation path to the PATH environment variable:

```
export PATH=/path/to/XHMM:SPATH
```

-----End
4.23.7 Running and Verifying XHMM

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Download the test case package `POST_GATK_DATA.zip`.

**Step 3** Use the SFTP tool to upload the test case package to the `/path/to/TEST` directory on the server.

**Step 4** Run the following command to decompress the test case package:

```shell
cd /path/to/TEST
unzip POST_GATK_DATA.zip
```

**Step 5** Run the following command to merge multiple samples (at the same point) output by GATK depths using XHMM:

```shell
xhmm --mergeGATKdepths -o ./DATA.RD.txt \
   --GATKdepths group1.DATA.sample_interval_summary \
   --GATKdepths group2.DATA.sample_interval_summary \
   --GATKdepths group3.DATA.sample_interval_summary
```

**Execution result:**

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Command-line parameter values:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mergeGATKdepths
GATKdepths="group1.DATA.sample_interval_summary"
GATKdepths="group2.DATA.sample_interval_summary"
GATKdepths="group3.DATA.sample_interval_summary"
outputMatrix="/DATA.RD.txt"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Reading GATK depth file group1.DATA.sample_interval_summary
Reading GATK depth file group2.DATA.sample_interval_summary
Reading GATK depth file group3.DATA.sample_interval_summary
Writing GATK read-depth matrix of 30 samples by 300 targets to ./DATA.RD.txt

Total CPU time for processing this job: 6 seconds
 [user time: 6  
system time: 0
children user time: 0
children system time: 0] 
Total time for processing this job: 0 seconds

Generated file:

DATA.RD.txt
```

----End

4.24 pysam 0.15.4 Porting Guide (CentOS 7.6)
4.24.1 Introduction

Pysam is a Python module used to read and operate files in SAM, BAM, VCF, and BCF formats. It is a lightweight wrapper of the htslib C-API and provides support for SAMtools, BCFtools, and tabix.

For more information about pysam, visit https://pysam.readthedocs.io/en/latest/api.html#.

Programming language: Python, Cython

Open-source license: MIT license

Brief description: A Python module used to read and operate SAM/BAM/VCF/BCF files.

Recommended Version

pysam-v0.15.4

4.24.2 Environment Requirements

Hardware Requirements

Table 4-121 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-122 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>pysam</td>
<td>pysam-v0.15.4</td>
<td><a href="https://github.com/pysam-developers/pysam/archive/v0.15.4.tar.gz">https://github.com/pysam-developers/pysam/archive/v0.15.4.tar.gz</a></td>
</tr>
<tr>
<td>Python</td>
<td>Python-3.8.2</td>
<td><a href="https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tar.xz">https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tar.xz</a></td>
</tr>
<tr>
<td>Cython</td>
<td>Cython 0.29.19</td>
<td><a href="https://files.pythonhosted.org/packages/79/36/69246177114d0b6cb7bd4f9aef177b434c0f4a767e05201b373e8c8d7092/Cython-0.29.19.tar.gz">https://files.pythonhosted.org/packages/79/36/69246177114d0b6cb7bd4f9aef177b434c0f4a767e05201b373e8c8d7092/Cython-0.29.19.tar.gz</a></td>
</tr>
</tbody>
</table>
OS Requirements

Table 4-123 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.24.3 Planning the Paths for Software Porting

Table 4-124 lists the software installation paths involved in the pysam software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see Planning the Installation Paths in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/PYSAM</td>
<td>Installation path of pysam</td>
<td>The installation path listed in this table is only an example. A shared path is recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/PYTHON</td>
<td>Installation path of Python</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/CYTHON</td>
<td>Installation path of Cython</td>
<td></td>
</tr>
</tbody>
</table>

4.24.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.
Configuration Process

Table 4-125 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Operation</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>Setting Up the Cluster Environment in <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>Install the Python interpreter.</td>
<td>4.24.4.1 Installing the Python Interpreter</td>
</tr>
<tr>
<td>3</td>
<td>Install the Cython package.</td>
<td>4.24.4.2 Installing the Cython Package</td>
</tr>
</tbody>
</table>

4.24.4.1 Installing the Python Interpreter

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
tar xf Python-3.8.2.tar.xz
```

Step 3 Switch to the directory in which the decompressed files are stored.

```
cd Python-3.8.2
```

Step 4 Create the makefile.

```
./configure --prefix=/path/to/PYTHON
```

Step 5 Run the following commands to compile and install the Python interpreter:

```
make -j 96
make install
```

Step 6 Set the Python environment variables.

```
export PATH=/path/to/PYTHON/bin:$PATH
```

---End

4.24.4.2 Installing the Cython Package

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Decompress the installation package.

```
tar xzvf Cython-0.29.19.tar.gz
```
Step 3  Switch to the directory in which the decompressed files are stored.
    cd Cython-0.29.19/
Step 4  Run the following command to compile Cython:
    python3 setup.py build
Step 5  Run the following command to install Cython:
    python3 setup.py install
    ----End

4.24.5 Obtaining the Source Code

Procedure

Step 1  Download the pysam source code package pysam-0.15.4.tar.gz.
    URL: https://github.com/pysam-developers/pysam/archive/v0.15.4.tar.gz
Step 2  Use the SFTP tool to upload the pysam installation package to the /path/to/PYSAM directory on the server.
    ----End

4.24.6 Compiling and Installing Pysam

Procedure

Step 1  Use PuTTY to log in to the server as the root user.
Step 2  Decompress the installation package.
    tar xzvf pysam-0.15.4.tar.gz
Step 3  Switch to the directory in which the decompressed files are stored.
    cd pysam-0.15.4/
Step 4  Run the following command to compile pysam:
    python3 setup.py build
Step 5  Run the following command to install pysam:
    python3 setup.py install
    ----End

4.24.7 Running and Verifying Pysam

Procedure

Step 1  Use PuTTY to log in to the server as the root user.
Step 2  Switch to the pysam-0.15.4/tests/pysam_data/ folder in the Pysam source code directory and locate example_aligned_pairs.sam.
Step 3  Run the following commands to run pysam:

```python
python3
import pysam

samfile = pysam.AlignmentFile("example_aligned_pairs.sam")

samfile.count()
```

In the command output, 52 indicates that the pysam successfully parses the SAM file.

---End

4.24.8 Troubleshooting

Issue 1

**Symptom**

During the installation and compilation of Python-3.8.2, the error message "ModuleNotFoundError: No module named '_ctypes'" is displayed.

**Possible Causes**

CentOS 7 is not installed with the development link library software package of the external function library (libffi).

**Procedure**

Before installing and compiling Python-3.8.2, run the `yum install libffi-devel -y` command to install the libffi-devel package.

Issue 2

**Symptom**

During the configuration process, the error message "configure: error: liblzma development files not found" is displayed.

**Possible Causes**

The dependency package `xz-devel` is missing.

**Procedure**

Before installing and compiling Python-3.8.2, run the `yum install xz-devel -y` command to install the `xz-devel` package.
4.24.9 More Information

https://pysam.readthedocs.io/en/latest/api.html#

4.25 Abyss 2.2.4 Porting Guide (CentOS 7.6)

4.25.1 Introduction

Abyss is a de novo, parallel, paired-end sequence assembler that is designed for short reads. The single-processor version is useful for assembling genomes up to 100 Mbases in size. The parallel version is implemented using MPI and is capable of assembling larger genomes. Mainstream NGS genome assembly software divides sequences into k-mers and then obtains assembled sequences based on the de Bruijn Graph algorithm. When the program is running, the k-mer strings are stored in the memory. Therefore, the memory of the computer must be large enough. Abyss is based on the Bloom filter. Its advantage is that it can perform parallel operations and run multiple stitching tasks at the same time. Therefore, Abyss processes more genes than NGS.

For more information about VarScan, visit the official Abyss website.

Programming language: C

One sentence description: a gene sequence assembler.

Open-source protocol: GPL

Recommended Version

Abyss-2.2.4

4.25.2 Environment Requirements

Hardware Requirements

Table 4-126 lists the hardware requirements.

Table 4-126 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table lists the software requirements.
Table 4-127 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abyss</td>
<td>Abyss-2.2.4</td>
<td><a href="https://github.com/bcgsc/abyss/archive/2.2.4.tar.gz">https://github.com/bcgsc/abyss/archive/2.2.4.tar.gz</a></td>
</tr>
<tr>
<td>boost</td>
<td>Boost-1.56.0</td>
<td><a href="http://downloads.sourceforge.net/project/boost/boost/1.56.0/boost_1_56_0.tar.bz2">http://downloads.sourceforge.net/project/boost/boost/1.56.0/boost_1_56_0.tar.bz2</a></td>
</tr>
<tr>
<td>test-data</td>
<td>Test case</td>
<td><a href="https://www.bcgsc.ca/platform/bioinfo/software/abyss/releases/1.3.4/test-data.tar.gz">https://www.bcgsc.ca/platform/bioinfo/software/abyss/releases/1.3.4/test-data.tar.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-128 lists the OS requirements.

Table 4-128 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.25.3 Paths for Software Porting

This section describes the software installation paths involved in the Abyss software porting.

Table 4-129 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment</td>
<td>For details, see section &quot;Planning Data for Installation&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Usage</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------------</td>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/ABYSS</td>
<td>Installation path of Abyss</td>
<td>The installation path listed in this table is only an example. A shared path is recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/BOOST</td>
<td>Installation path of boost</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>/path/to/TESTCASE</td>
<td>Installation path of Abyss</td>
<td></td>
</tr>
</tbody>
</table>

### 4.25.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.

**Configuration Process**

**Table 4-130** Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Cluster Environment&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
</tbody>
</table>

### 4.25.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the Abyss installation package `abyss-2.2.4.tar.gz`.

Download address: [https://github.com/bcgsc/abyss/archive/2.2.4.tar.gz](https://github.com/bcgsc/abyss/archive/2.2.4.tar.gz)

**Step 2** Download the dependency `boost_1_56_0.tar.bz2`.

Download address: [http://downloads.sourceforge.net/project/boost/boost/1.56.0/boost_1_56_0.tar.bz2](http://downloads.sourceforge.net/project/boost/boost/1.56.0/boost_1_56_0.tar.bz2)

**Step 3** Download the test case package `test-data.tar.gz`.

Download address: [https://www.bcgsc.ca/platform/bioinfo/software/abyss/releases/1.3.4/test-data.tar.gz](https://www.bcgsc.ca/platform/bioinfo/software/abyss/releases/1.3.4/test-data.tar.gz)

**Step 4** Use an SFTP tool to upload the Abyss installation package to the `/path/to/ABYSS` directory on the server.

**Step 5** Use an SFTP tool to upload the Boost installation package to the `/path/to/BOOST` directory on the server.
Step 6 Use an SFTP tool to upload the Abyss test case package to the `/path/to/TESTCASE` directory on the server.

---End

4.25.6 Compiling and Installing Abyss

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following commands to decompress the installation package:

```
tar -zxvf abyss-2.2.4.tar.gz
```

Step 3 Run the following commands to decompress the dependency package:

```
cd /path/to/BOOST

tar jxf boost_1_56_0.tar.bz2
```

Step 4 Switch to the folder generated after the package is decompressed.

```
cd /path/to/ABYSS/abyss-2.2.4
```

Step 5 Run the following command to generate a configure file:

```
./autogen.sh
```

Step 6 Run the following command to add the installation path of the currently running MPI to the temporary environment variables:

```
export CURRENT_MPI_PATH=`which mpirun | sed 's//bin/mpirun//g'`
```

Step 7 Run the following command to perform configuration:

```
./configure --prefix=/path/to/ABYSS --with-boost=/path/to/BOOST/boost_1_56_0 --without-sparsehash --with-mpi=$CURRENT_MPI_PATH
```

Step 8 Run the following commands to compile and install Abyss:

```
make -j 32 AM_CXXFLAGS=-Wall
make install AM_CXXFLAGS=-Wall
```

Step 9 Run the following commands to add the environment variables:

```
export PATH=/path/to/ABYSS/bin:$PATH
```

---End

4.25.7 Running and Verifying Abyss

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Run the following command to decompress the test case package:

```
tar xvf test-data.tar.gz
```
Step 3  Switch to the directory and run the following command to assemble Abyss:

```
  cd /path/to/TESTCASE
  abyss-pe k=25 name=test in='reads1.fastq reads2.fastq'
```

Step 4  After the preceding steps are performed, only a single process can be executed. If multiple processes are executed, an error is reported, as shown in the following figure. To execute multiple processes concurrently, perform the following steps to modify the file.

1. Switch to the Abyss installation path and find the `abyss-pe` file.
```
  cd /path/to/ABYSS/bin
```

2. Modify the `abyss-pe` file.
```
  vi abyss-pe
  Press i to enter the editing mode and add `--allow-run-as-root -mca
coll_hcoll_enable 0` to lines 543 and 551 of `abyss-pe`. After the modification, the content is as follows:

```
$(gtime) $(mpirun) --allow-run-as-root -mca coll_hcoll_enable 0 -np $(np) abyss-paired-dbg-mpi $(abyssopt) $(ABYSS_OPTIONS) -o $*-1.fa $(in) $(se)
$(gtime) $(mpirun) --allow-run-as-root -mca coll_hcoll_enable 0 -np $(np) ABYSS-P $(abyssopt) $(ABYSS_OPTIONS) -o $@ $(in) $(se)
```

Press Esc, enter `:wq!`, and press Enter to save the file and exit.

3. Switch the directory and perform Abyss assembly in parallel.
```
  cd /path/to/TESTCASE/
  abyss-pe np=96 k=25 name=test in='reads1.fastq reads2.fastq'
```

**NOTE**

The `name` parameter indicates the prefix of the generated file. After the execution is complete, many files are generated, among which `test-contigs.fa` and `test-scaffolds.fa` correspond to the `contig` and `scaffold` results, respectively.

Results:

<table>
<thead>
<tr>
<th>n</th>
<th>n:1000</th>
<th>L50</th>
<th>min</th>
<th>N80</th>
<th>N50</th>
<th>N20</th>
<th>E-size</th>
<th>max</th>
<th>sum</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>756</td>
<td>19</td>
<td>6</td>
<td>1315</td>
<td>1752</td>
<td>3094</td>
<td>4947</td>
<td>3141</td>
<td>6122</td>
<td>47778</td>
<td>10</td>
</tr>
<tr>
<td>757</td>
<td>20</td>
<td>7</td>
<td>1315</td>
<td>1700</td>
<td>2214</td>
<td>4947</td>
<td>2991</td>
<td>6122</td>
<td>47778</td>
<td>20</td>
</tr>
<tr>
<td>759</td>
<td>22</td>
<td>9</td>
<td>1315</td>
<td>1700</td>
<td>2205</td>
<td>3191</td>
<td>2385</td>
<td>4066</td>
<td>47789</td>
<td>50</td>
</tr>
<tr>
<td>759</td>
<td>22</td>
<td>9</td>
<td>1315</td>
<td>1700</td>
<td>2205</td>
<td>3191</td>
<td>2385</td>
<td>4066</td>
<td>47789</td>
<td>100</td>
</tr>
</tbody>
</table>
Best scaffold N50 is 3094 at n=10 s=1000.

<table>
<thead>
<tr>
<th>n</th>
<th>n:1000</th>
<th>L50</th>
<th>min</th>
<th>N80</th>
<th>N50</th>
<th>N20</th>
<th>E-size</th>
<th>max</th>
<th>sum</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>756</td>
<td>19</td>
<td>6</td>
<td>1315</td>
<td>1752</td>
<td>3094</td>
<td>4947</td>
<td>3141</td>
<td>6122</td>
<td>47778</td>
<td>n=10 s=1000</td>
</tr>
</tbody>
</table>

PathConsensus --dot -k25 -p0.9 -s test-7.fa -g test-7.dot -o test-7.path test-6.fa test-6.dot test-6.path

Ambiguous paths: 3
Merged: 0
No paths: 3
Too many paths: 0
Too complex: 0
Dissimilar: 0
cat test-6.fa test-7.fa \
|MergeContigs -k25 -o test-8.fa - test-7.dot test-7.path

The minimum coverage of single-end contigs is 2.28.
The minimum coverage of merged contigs is 3.83248.
Consider increasing the coverage threshold parameter, c, to 3.83248.
In -sf test-8.fa test-scaffolds.fa
PathOverlap --overlap -k25 --dot test-7.dot test-7.path >test-8.dot
In -sf test-8.dot test-scaffolds.dot
abyss-fac test-unitigs.fa test-contigs.fa test-scaffolds.fa | tee test-stats.tab

n   n:500   L50   min   N80   N50   N20   E-size max sum name

----End

4.26 Chaste 2019.1 Porting Guide (CentOS 7.6)

4.26.1 Introduction

Chaste (Cancer, Heart and Soft Tissue Environment) is a general purpose simulation package aimed at multi-scale, computationally demanding problems arising in biology and physiology. Current functionality includes tissue and cell level electrophysiology, discrete tissue modelling, and soft tissue modelling. The package is being developed by a team mainly based in the Computational Biology Group at the Department of Computer Science, University of Oxford, and development draws on expertise from software engineering, high performance computing, mathematical modelling, and scientific computing.

Programming language: C++
Brief description: general purpose simulation package
Open-source protocol: 3-clause BSD

Recommended Version

CHASTE 2019.1

4.26.2 Environment Requirements

Hardware Requirements

Table 4-131 lists the hardware requirements.
Table 4-131 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table Software Requirements lists the software requirements.

Table 4-132 Software Requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOST</td>
<td>1.58.0</td>
<td><a href="http://downloads.sourceforge.net/project/boost/boost/1.58.0/boost_1_58_0.tar.gz">http://downloads.sourceforge.net/project/boost/boost/1.58.0/boost_1_58_0.tar.gz</a></td>
</tr>
<tr>
<td>HDF5</td>
<td>1.8.16</td>
<td><a href="https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8.16/src/hdf5-1.8.16.tar.bz2">https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8.16/src/hdf5-1.8.16.tar.bz2</a></td>
</tr>
<tr>
<td>Python</td>
<td>2.5</td>
<td><a href="https://www.python.org/ftp/python/2.5/Python-2.5.tgz">https://www.python.org/ftp/python/2.5/Python-2.5.tgz</a></td>
</tr>
<tr>
<td>Amara</td>
<td>1.2.0.2</td>
<td><a href="https://files.pythonhosted.org/packages/af/0e/e81dfe4b4fb24023207d86e4d4ff418b0e83927f2155ed3552b0e3196846/Amara-1.2.0.2.tar.gz">https://files.pythonhosted.org/packages/af/0e/e81dfe4b4fb24023207d86e4d4ff418b0e83927f2155ed3552b0e3196846/Amara-1.2.0.2.tar.gz</a></td>
</tr>
<tr>
<td>4Suite-XML</td>
<td>1.0.2</td>
<td><a href="https://files.pythonhosted.org/packages/0e/a8/3c5b4fffb12be7c3a80c99475853349e1cf8477f99051921ea06fbf5e3b9/4Suite-XML-1.0.2.tar.gz">https://files.pythonhosted.org/packages/0e/a8/3c5b4fffb12be7c3a80c99475853349e1cf8477f99051921ea06fbf5e3b9/4Suite-XML-1.0.2.tar.gz</a></td>
</tr>
<tr>
<td>lxml</td>
<td>3.2.1</td>
<td><a href="https://files.pythonhosted.org/packages/a8/17/942c2f167cade0f387b2077299865b6e1d26dca75e1587e12df408b67d9a/lxml-3.2.1.tar.gz">https://files.pythonhosted.org/packages/a8/17/942c2f167cade0f387b2077299865b6e1d26dca75e1587e12df408b67d9a/lxml-3.2.1.tar.gz</a></td>
</tr>
<tr>
<td>rdflib</td>
<td>2.4.2</td>
<td><a href="https://files.pythonhosted.org/packages/04/06/3904f93f05a4734e4daec069aba9d590f53db8990a81e8e79fffe6602466/rdflib-2.4.2.tar.gz">https://files.pythonhosted.org/packages/04/06/3904f93f05a4734e4daec069aba9d590f53db8990a81e8e79fffe6602466/rdflib-2.4.2.tar.gz</a></td>
</tr>
<tr>
<td>python-dateutil</td>
<td>1.5</td>
<td>CentOS 7.6 software package</td>
</tr>
</tbody>
</table>
### Item | Version | How to Obtain
--- | --- | ---
xsd | 4.0.0-25 | https://mirrors.tuna.tsinghua.edu.cn/epel/7/aarch64/Packages/x/xsd-4.0.0-25.el7.aarch64.rpm
xerces | 3.1.1-10 | CentOS 7.6 software package
vtk | 6.2.0 | http://www.vtk.org/files/release/6.2/VTK-6.2.0.tar.gz
metis | 5.1.0 | http://ftp.mcs.anl.gov/pub/petsc/externalpackages/metis-5.1.0-p1.tar.gz
parmetis | 4.0.3 | http://ftp.mcs.anl.gov/pub/petsc/externalpackages/parmetis-4.0.3-p2.tar.gz
f2cblaslapack | 3.4.2 | http://ftp.mcs.anl.gov/pub/petsc/externalpackages/f2cblaslapack-3.4.2.q1.tar.gz
sundials | 2.5.0 | http://ftp.mcs.anl.gov/pub/petsc/externalpackages/sundials-2.5.0p1.tar.gz
setuptools | 0.6c8 | https://files.pythonhosted.org/packages/29/17/f98a2cb39bb5b40357fd54fba1dd105e224a47a91867ac5c4ef68f0191/setuptools-0.6c8.tar.gz
OpenBLAS | 0.3.6 | https://github.com/xianyi/OpenBLAS/tree/v0.3.6
Test case | - | https://chaste.cs.ox.ac.uk/chaste/tutorials/release_3.3/UserTutorials/CardiacExecutable/Propagation3d.tgz

### OS Requirements

**Table 4-133** lists the OS requirements.

**Table 4-133 OS requirements**

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.26.3 Paths for Software Porting

This section describes the software installation paths involved in the CHASTE software porting.
### Table 4-134 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see section “Planning Data for Installation” in <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/CHASTE</td>
<td>Installation path of CHASTE.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>/path/to/PETSC</td>
<td>Installation path of PETSC.</td>
<td>The installation path listed in this table is only an example. A shared path is recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>4</td>
<td>path/to/BOOST</td>
<td>Installation path of BOOST.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>path/to/PYTHON</td>
<td>Installation path of Python.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>path/to/AMARA</td>
<td>Installation path of Amara.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>path/to/LXML</td>
<td>Installation path of lxml.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>path/to/RDFLIB</td>
<td>Installation path of rdflib.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>path/to/VTK</td>
<td>Installation path of VTK.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>path/to/XSD</td>
<td>Installation path of XSD.</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS.</td>
<td></td>
</tr>
</tbody>
</table>

### 4.26.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.
Configuration Process

Table 4-135 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Cluster Environment&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing xerces</td>
<td>For details, see 4.26.4.1 Installing xerces and python-dateutil.</td>
</tr>
<tr>
<td>3</td>
<td>Installing python-dateutil</td>
<td>For details, see 4.26.4.1 Installing xerces and python-dateutil.</td>
</tr>
<tr>
<td>4</td>
<td>Installing PETSC</td>
<td>For details, see 4.26.4.2 Installing PETSC.</td>
</tr>
<tr>
<td>5</td>
<td>Installing BOOST</td>
<td>For details, see 4.26.4.3 Installing BOOST.</td>
</tr>
<tr>
<td>6</td>
<td>Installing Python</td>
<td>For details, see 4.26.4.4 Installing Python.</td>
</tr>
<tr>
<td>7</td>
<td>Installing Amara</td>
<td>For details, see 4.26.4.5 Installing Amara.</td>
</tr>
<tr>
<td>8</td>
<td>Installing XSD</td>
<td>For details, see 4.26.4.6 Installing xsd.</td>
</tr>
<tr>
<td>9</td>
<td>Installing rdflib</td>
<td>For details, see 4.26.4.7 Installing rdflib.</td>
</tr>
<tr>
<td>10</td>
<td>Installing VTK</td>
<td>For details, see 4.26.4.8 Installing vtk.</td>
</tr>
<tr>
<td>11</td>
<td>Installing lxml</td>
<td>For details, see 4.26.4.9 Installing lxml.</td>
</tr>
<tr>
<td>12</td>
<td>Installing OpenBLAS</td>
<td>For details, see 4.26.4.10 Installing OpenBLAS.</td>
</tr>
</tbody>
</table>

4.26.4.1 Installing xerces and python-dateutil

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to install xerces and python-dateutil:

```
yum install xerces-c.aarch64 xerces-c-devel.aarch64 python-lxml.aarch64 python-dateutil.noarch -y
```

----End
4.26.4.2 Installing PETSC

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the metis, parmetis, f2cblaslapack, and sundials software packages to the /path/to/PETSC directory.

Step 3 Run the following commands to decompress the PETSC installation package:

```
cd /path/to/PETSC
tar -xvf petsc-lite-3.6.2.tar.gz
```

Step 4 Switch to the folder generated after the package is decompressed.

```
cd petsc-3.6.2
```

Step 5 Run the following commands to load environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 6 Run the following command to perform configuration:

```
./configure --with-make-np=4 --with-fc=0 --with-x=false --with-ssl=false --download-f2cblaslapack=/path/to/PETSC/f2cblaslapack-3.4.2.q1.tar.gz --download-parmetis=/path/to/PETSC/parmetis-4.0.3-p2.tar.gz --download-sundials=/path/to/PETSC/sundials-2.5.0p1.tar.gz --with-shared-libraries --with-mpi-dir=$MPI_DIR --download-metis=/path/to/PETSC/metis-5.1.0-p1.tar.gz --download-hdf5=/path/to/PETSC/hdf5-1.8.16.tar.bz2
```

**NOTE**

In this example, the --download parameter specifies other software packages to be installed together.

The --download parameter can be in the format of --download-hdf5=1 or a specified path. In the format of --download-hdf5=1, the software package is downloaded from the network and then installed. In the format of a specific path, the software package in the path is directly installed.

1. If an error is reported in 6, run the following commands to search for the config.guess and config.sub files:

```
find ./ -name config.guess
find ./ -name config.sub
```

2. Update the content in the files to the content on the web page.

config.guess:

```
http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD
```

config.sub:

```
http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD
```
3. After the installation is complete, repeat Step 6.

**Step 7** Run the following command to install the software:

```make```
```
```

**Step 8** Run the following commands to set environment variables:

```bash
export PETSC_DIR=/path/to/PETSC/petsc-3.6.2
export PETSC_ARCH=linux-gnu
export SUNDIALS_ROOT=$PETSC_DIR/$PETSC_ARCH
export HDF5_ROOT=$PETSC_DIR/$PETSC_ARCH
```

----End

### 4.26.4.3 Installing BOOST

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Upload the BOOST source code package to the /path/to/BOOST directory.

**Step 3** Decompress the BOOST installation package.

```bash
cd /path/to/BOOST
tar -xvf boost_1_58_0.tar.gz
```

**Step 4** Switch to the folder generated after the package is decompressed.

```bash
cd boost_1_58_0
```

**Step 5** Run the following command to modify the source code of BOOST.

```bash
sed -ri 's/\-m64/\-mabi=lp64/g' `grep -Rl '\-m64'`
```

**Step 6** Run the following command to perform configuration:

```bash
./bootstrap.sh --prefix=/path/to/BOOST/boost_1_58_0/build --with-libraries=system,filesystem,serialization,program_options --with-toolset=gcc
```

**Step 7** Run the following commands to compile and install BOOST:

```bash
./b2 install
```

**Step 8** Run the following command to set the BOOST environment variable:

```bash
export LD_LIBRARY_PATH=/path/to/BOOST/lib:SLD_LIBRARY_PATH
```

----End

### 4.26.4.4 Installing Python

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.
Step 2 Upload the Python source code package to the path/to/PYTHON directory.

Step 3 Run the following commands to decompress the Python installation package:

```bash
cd /path/to/PYTHON

 tar -zxvf Python-2.5.tgz
```

Step 4 Switch to the folder generated after the package is decompressed.

```bash
cd Python-2.5
```

Step 5 Run the following command to perform configuration:

```bash
SVNVERSION=not-found ./configure --prefix=path/to/PYTHON
```

Step 6 Run the following commands to compile and install Python:

```bash
make -j 16
make install
```

Step 7 Run the following command to set Python environment variable:

```bash
export PATH=path/to/PYTHON/bin:$PATH
```

----End

4.26.4.5 Installing Amara

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the 4Suite-XML and Amara source code packages to the /path/to/AMARA directory.

Step 3 Run the following commands to install the dependency 4Suite-XML-1.0.2:

```bash
cd /path/to/AMARA

tar -xvf 4Suite-XML-1.0.2.tar.gz

cd 4Suite-XML-1.0.2

python setup.py install
```

Step 4 Run the following commands to decompress the Amara installation package:

```bash
cd /path/to/AMARA

tar -zxvf Amara-1.2.0.2.tar.gz
```

Step 5 Switch to the folder generated after the package is decompressed.

```bash
cd Amara-1.2.0.2
```

Step 6 Run the following command to perform installation:

```bash
python setup.py install
```
The Amara version must be 1.2.0.2, because 2.* versions are incompatible with CHASTE. You need to use Python 2.5 to install Amara 1.2.0.2.

---End

4.26.4.6 Installing xsd

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the XSD software package to the /path/to/XSD directory.

Step 3 Run the following commands to install the XSD software package:

```
cd path/to/XSD
rpm -ivh xsd-4.0.0-25.el7.aarch64.rpm
```

- NOTE
  If the network Yum source can be configured, you can use the yum install command to install XSD.
  If an error message is displayed indicating that the dependency is missing during manual downloading and installing the RPM package, download and install the dependency RPM package.

---End

4.26.4.7 Installing rdflib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Upload the RDFLib and setuptools source code packages to the /path/to/RDFLIB directory.

Step 3 Run the following command to install the dependency setuptools:

```
cd /path/to/RDFLIB
unzip setuptools-0.6c8.zip
cd setuptools-0.6c8
python setup.py install
```

Step 4 Run the following commands to decompress the RDFLib installation package:

```
cd path/to/RDFLIB
tar -zxvf rdflib-2.4.2.tar.gz
```

Step 5 Switch to the folder generated after the package is decompressed.

```
cd rdflib-2.4.2
```
Step 6  Run the following commands to perform installation:

```
yum install libxml2-devel libxslt-devel -y
python setup.py install
```
-----End

4.26.4.8 Installing vtk

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Upload the VTK source code package to the /path/to/VTK directory.

Step 3  Run the following commands to decompress the VTK installation package:

```
cd path/to/VTK
.tar -xvf VTK-6.2.0.tar.gz
.cd VTK-6.2.0
```

Step 4  Run the following commands to compile and install VTK:

```
cmake -DCMAKE_INSTALL_PREFIX=path/to/VTK
make -j 16
make install
```

Step 5  Run the following command to set the VTK environment variable:

```
export PATH=/path/to/VTK/bin:SPATH
```
-----End

4.26.4.9 Installing lxml

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Upload the lxml source code package to the path/to/LXML directory.

Step 3  Decompress the lxml source code package.

```
cd /path/to/LXML
.tar -xvf lxml-3.2.1.tar.gz
```

Step 4  Run the following commands to install lxml:

```
cd lxml-3.2.1
python setup.py install
```
-----End
### 4.26.4.10 Installing OpenBLAS

**Procedure**

**Step 1** Use PuTTY to log into the server as the **root** user.

**Step 2** Upload the OpenBLAS source code package to the *path/to/OPENBLAS* directory.

**Step 3** Run the following command to install BLAS and LAPACK:

```
yum install blas64.aarch64 blas64-devel.aarch64 lapack64.aarch64 lapack64-devel.aarch64 -y
```

**Step 4** Decompress the OpenBLAS source code package.

```
cd /path/to/OPENBLAS
tar -xvf OpenBLAS-0.3.6.tar.gz
```

**Step 5** Switch to the folder generated after the package is decompressed.

```
cd OpenBLAS-0.3.6
```

**Step 6** Run the following commands to compile and install OpenBLAS:

```
make
make PREFIX=/path/to/OPENBLAS install
```

----End

### 4.26.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the **Chaste-release_2019.1.tar.gz** source code package.


**Step 2** Use the SFTP tool to upload the CHASTE source code package to the */path/to/CHASTE* directory on the server.

----End

### 4.26.6 Compiling and Installing CHASTE

**Procedure**

**Step 1** Use PuTTY to log into the server as the **root** user.

**Step 2** Decompress the source code package.

```
cd /path/to/CHASTE
tar -xvf Chaste-release_2019.1.tar.gz
```

**Step 3** Run the following command to modify the **translators.py** file.

```
vim /path/to/CHASTE/Chaste-release_2019.1/python/pycml/translators.py
```
Uncomment line 1830 and comment out line 1831.

Uncomment line 1838 and comment out line 1839.

Before the modification:

```cpp
1830                 #self.writeln('std::cout << "Too small: ', self.code_name(var), ' = " << ', self.code_name(var) , ' << std::endl << std::flush;')
1831                 self.writeln(error_template.format(self.var_display_name(var)))
1832                 self.close_block(False)
1833             for var in high_range_vars:
1834                 if using_cvode:
1835                     additional_tolerance_adjustment = ' + tol'
1836                     self.writeln('if (' , self.code_name(var), ' > ', var.get_rdf_annotation(high_prop),
additional_tolerance_adjustment, ')')
1837                 self.open_block()
1838                 #self.writeln('std::cout << "Too large: ', self.code_name(var), ' = " << ', self.code_name(var) , ' << std::endl << std::flush;')
1839                 self.writeln(error_template.format(self.var_display_name(var)))
```

After the modification:

```cpp
1830                 self.writeln('std::cout << "Too small: ', self.code_name(var), ' = " << ', self.code_name(var) , ' << std::endl << std::flush;')
1831                 #self.writeln(error_template.format(self.var_display_name(var)))
1832                 self.close_block(False)
1833             for var in high_range_vars:
1834                 if using_cvode:
1835                     additional_tolerance_adjustment = ' + tol'
1836                     self.writeln('if (' , self.code_name(var), ' > ', var.get_rdf_annotation(high_prop),
additional_tolerance_adjustment, ')')
1837                 self.open_block()
1838                 self.writeln('std::cout << "Too large: ', self.code_name(var), ' = " << ', self.code_name(var) , ' << std::endl << std::flush;')
1839                 #self.writeln(error_template.format(self.var_display_name(var)))
```

**Step 4** Run the following commands to compile and install CHASTE:

```
cd /path/to/CHASTE
mkdir chaste_build
cd chaste_build
cmake -DCMAKE_INSTALL_PREFIX=/path/to/CHASTE/chaste_build /path/to/CHASTE/Chaste-release_2019.1 -DBOOST_INCLUDE_DIR=/path/to/CHASTE/Chaste-release_2019.1 -DBOOST_LIBS/boost_1_58_0/build/include
make -j48 Continuous
make install
export PATH=/path/to/CHASTE/chaste_build/apps:$PATH
```

**4.26.7 Running and Verifying CHASTE**

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to switch to the CHASTE folder:

```bash
cd /path/to/CHASTE
```
Step 3 Run the following commands to run the test case:

```
tar -xvf Propagation3d.tgz
```

```
cd Propagation3d
```

```
Chaste ChasteParameters.xml
```

Step 4 After the execution is complete, the following information is displayed and the corresponding file is generated in the `/tmp/root/testoutput/ChasteResults` directory:

<table>
<thead>
<tr>
<th>InMesh</th>
<th>Init</th>
<th>AssSys</th>
<th>Ode</th>
<th>Comms</th>
<th>AssRhs</th>
<th>NeuBCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirBCs</td>
<td>Ksp</td>
<td>Output</td>
<td>DataConversion</td>
<td>PostProc</td>
<td>User1</td>
<td>User2</td>
</tr>
<tr>
<td>User3</td>
<td></td>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>24.748 (17%)</td>
<td>15.570 (10%)</td>
<td>76.167 (51%)</td>
<td>23.206 (15%)</td>
<td>0.032 (0%)</td>
<td>0.605 (0%)</td>
</tr>
</tbody>
</table>

**NOTE**

- `progress_status.txt`: file you can use to estimate the required time when performing the simulation.
- `3dResults.h5`: file that stores analog output in HDF5 format.
- `Output`: folder that stores the output in Meshalyzer format.
- `cmgui_output`: folder that stores the output in the CMGUI format.
- `vtk_output`: folder that stores the output in Paraview (VTK) format.

Step 5 Run the following command to run the test case:

```
mpirun -np 96 --mca btl ^openib --allow-run-as-root Chaste ChasteParameters.xml
```

Step 6 After the execution is complete, the following information is displayed and the corresponding file is generated in the `/tmp/root/testoutput/ChasteResults` directory:

<table>
<thead>
<tr>
<th>Proc</th>
<th>InMesh</th>
<th>Init</th>
<th>AssSys</th>
<th>Ode</th>
<th>Comms</th>
<th>AssRhs</th>
<th>NeuBCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>User2</td>
<td></td>
<td>User3</td>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.108 (21%)</td>
<td>0.676 (3%)</td>
<td>0.881 (4%)</td>
<td>0.355 (1%)</td>
<td>0.061 (0%)</td>
<td>0.015 (0%)</td>
<td>0.000 (0%)</td>
</tr>
<tr>
<td>1:</td>
<td>5.195 (21%)</td>
<td>0.759 (3%)</td>
<td>1.147 (5%)</td>
<td>0.356 (1%)</td>
<td>0.046 (0%)</td>
<td>0.015 (0%)</td>
<td>0.000 (0%)</td>
</tr>
<tr>
<td>2:</td>
<td>5.109 (21%)</td>
<td>0.676 (3%)</td>
<td>0.867 (4%)</td>
<td>0.355 (1%)</td>
<td>0.061 (0%)</td>
<td>0.014 (0%)</td>
<td>0.000 (0%)</td>
</tr>
<tr>
<td>3:</td>
<td>5.129 (21%)</td>
<td>0.692 (3%)</td>
<td>0.953 (4%)</td>
<td>0.355 (1%)</td>
<td>0.059 (0%)</td>
<td>0.014 (0%)</td>
<td>0.000 (0%)</td>
</tr>
<tr>
<td>4:</td>
<td>5.104 (21%)</td>
<td>0.672 (3%)</td>
<td>0.857 (3%)</td>
<td>0.356 (1%)</td>
<td>0.061 (0%)</td>
<td>0.015 (0%)</td>
<td>0.000 (0%)</td>
</tr>
</tbody>
</table>

**4.26.8 Troubleshooting**

**Problem 1: An Error Is Reported During PETSC Configuration**

**Symptom**
After the ./configure command is run, the following error information is displayed:

This script, last modified 2008-01-23, has failed to recognize
the operating system you are using. It is advised that you
download the most up to date version of the config scripts from
http://git.savannah.gnu.org/gitweb/?p=config.git;a=svn_plain;f=config.guess;hb=HEAD
and
http://git.savannah.gnu.org/gitweb/?p=config.git;a=svn_plain;f=config.sub;hb=HEAD

Possible Causes

The config.guess and config.sub certificate files have expired.

Procedure

Step 1 Run the following commands to search for the config.guess and config.sub files:

find . -name config.guess
find . -name config.sub

Step 2 Update the content in the files to the content in the web page.

config.guess:

http://git.savannah.gnu.org/gitweb/?p=config.git;a=svn_plain;f=config.guess;hb=HEAD

config.sub:

http://git.savannah.gnu.org/gitweb/?p=config.git;a=svn_plain;f=config.sub;hb=HEAD

-----End

Problem 2: An Error Is Reported During BOOST Installation

Symptom

After the ./b2 install command is run, the error message "gcc: error: unrecognized command line option '-m64'" is displayed.

Possible Causes

-m64 is an application compilation option for the x86 64-bit platform. To generate code for the AMD x86 64-bit architecture, set int to 32 bits and long, and pointer to 64 bits. However, the Arm 64-bit platform does not support this setting.

Procedure

Run the following command to modify the source code in BOOST and set the compilation option of the Arm 64-bit platform to -mabi=lp64:

sed -ri 's/\-m64/\-mabi=lp64/g' `grep -RL "\-m64`

Problem 3: An Error Is Reported During Python Configuration

Symptom

After the ./configure command is run, the error message "gcc: error: directory": No such file or directory" is displayed.
Possible Causes
A bug exists in the earlier Python version.

Procedure
Run the following command:

SVNVERSION=not-found ./configure

Problem 4: An Error Is Reported During CHASTE Installation

Symptom
After the make -j48 Continuous command is run, the error message "AttributeError: 'str' object has no attribute 'format'" is displayed.

Possible Causes
Format is a new method for formatting strings in Python 2.6.

Procedure
Run the following command to modify the translators.py file.

vim /path/to/CHASTE/Chaste-release_2019.1/python/pycml/translators.py

Uncomment line 1830 and comment out line 1831.
Uncomment line 1838 and comment out line 1839.

Before the modification:

```python
1830                 #self.writeln('std::cout << "Too small: ', self.code_name(var), ' = " << ', self.code_name(var) ,
' << std::endl << std::flush;')
1831                 self.writeln(error_template.format(self.var_display_name(var))))
1832                 self.close_block(False)
1833                 self.writeln('if (', self.code_name(var), ' > ', var.get_rdf_annotation(high_prop),
1834                 additional_tolerance_adjustment, ')')
1835                 self.open_block()
1836                 self.writeln('std::cout << "Too large: ', self.code_name(var), ' = " << ', self.code_name(var) ,
' << std::endl << std::flush;')
1837                 self.writeln(error_template.format(self.var_display_name(var)))
1838                 self.close_block(False)
1839                 for var in high_range_vars:
1840                 if using_cvode:
1841                 additional_tolerance_adjustment = ' + tol'
1842                 self.writeln('if (', self.code_name(var), ' > ', var.get_rdf_annotation(high_prop),
1843                 additional_tolerance_adjustment, ')')
1844                 self.open_block()
1845                 self.writeln('std::cout << "Too small: ', self.code_name(var), ' = " << ', self.code_name(var) ,
' << std::endl << std::flush;')
1846                 self.writeln(error_template.format(self.var_display_name(var)))
1847                 self.close_block(False)
1848                 for var in high_range_vars:
1849                 if using_cvode:
1850                 additional_tolerance_adjustment = ' + tol'
1851                 self.writeln('if (', self.code_name(var), ' > ', var.get_rdf_annotation(high_prop),
1852                 additional_tolerance_adjustment, ')')
1853                 self.open_block()
1854                 self.writeln('std::cout << "Too large: ', self.code_name(var), ' = " << ', self.code_name(var) ,
' << std::endl << std::flush;')
1855                 self.writeln(error_template.format(self.var_display_name(var)))
```

After the modification:

```python
1830                 self.writeln('std::cout << "Too small: ', self.code_name(var), ' = " << ', self.code_name(var) ,
' << std::endl << std::flush;')
1831                 self.writeln(error_template.format(self.var_display_name(var))))
1832                 self.close_block(False)
1833                 for var in high_range_vars:
1834                 if using_cvode:
1835                 additional_tolerance_adjustment = ' + tol'
1836                 self.writeln('if (', self.code_name(var), ' > ', var.get_rdf_annotation(high_prop),
1837                 additional_tolerance_adjustment, ')')
1838                 self.open_block()
1839                 self.writeln('std::cout << "Too large: ', self.code_name(var), ' = " << ', self.code_name(var) ,
' << std::endl << std::flush;')
```

4.26.9 More Information
For more information, visit the CHASTE official website:
4.27 COPASI 4.27.217 Porting Guide (CentOS 7.6)

4.27.1 Introduction

COPASI is a software application for simulation and analysis of biochemical networks and their dynamics. COPASI is a stand-alone program that supports models in the SBML standard and can simulate their behavior using ODEs or Gillespie's stochastic simulation algorithm; arbitrary discrete events can be included in such simulations.

For more information about COPASI, visit the official COPASI website.

Programming language: C++

Brief description: a biochemical system simulator.

Open-source protocol: Artistic License 2.0

Recommended Version

COPASI 4.27.217

4.27.2 Environment Requirements

Hardware Requirements

Table 4-136 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table Software Requirements lists the software requirements.

Table 4-137 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPASI</td>
<td>4.27.217</td>
<td><a href="https://codeload.github.com/copasi/COPASI/tar.gz/Build-217">https://codeload.github.com/copasi/COPASI/tar.gz/Build-217</a></td>
</tr>
<tr>
<td>Item</td>
<td>Version</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

### OS Requirements

**Table 4-138** lists the OS requirements.

**Table 4-138 OS requirements**

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.27.3 Planning the Path for Software Porting

This section describes the software installation paths involved in the COPASI software porting.

Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see section &quot;Planning Data for Installation&quot; in <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/COPASI</td>
<td>Installation path of COPASI.</td>
<td>The installation path listed in this table is only an example. A shared path is recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>
4.27.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see section &quot;Setting Up the Cluster Environment&quot; in <em>HPC Solution Basic Environment Setup Guide.</em></td>
</tr>
</tbody>
</table>

4.27.5 Obtaining the Source Code

Procedure

**Step 1** Download the COPASI installation package `COPASI-Build-217.tar.gz`.
Download address: [https://codeload.github.com/copasi/COPASI/tar.gz/Build-217](https://codeload.github.com/copasi/COPASI/tar.gz/Build-217)

**Step 2** Download the library installation package `copasi-dependencies-4.26.213.tar.gz` from the official website.

**Step 3** Use SFTP to upload the downloaded installation packages to the `/path/to/COPASI` directory on the server.

---End

4.27.6 Compiling and Installing COPASI

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the `yum` command to install the required dependency.

```
yum install qt.aarch64 qt-devel.aarch64 -y
```

**Step 3** Run the following commands to set environment variables:

```
export PATH=/path/to/cmake/bin:SPATH
export CC=`which gcc`
export CXX=`which g++`
```
Step 4 Run the following command to decompress the official dependency library installation package:
```bash
cd /path/to/COPASI
tar -xvf copasi-dependencies-4.26.213.tar.gz
```

Step 5 Run the following commands to compile and install the dependencies:
```bash
cd copasi-dependencies-4.26.213
./createLinux.sh
```

Step 6 Run the following command to copy the files in the `lib64` directory to the `lib` directory:
```bash
cp -r ./bin/lib64/* ./bin/lib
```

Step 7 Decompress the COPASI source code package.
```bash
cd /path/to/COPASI
tar -xvf COPASI-Build-217.tar.gz
```

Step 8 Run the following commands to compile and install COPASI:
```bash
cd COPASI-Build-217/
mkdir build
cd build
make -DBUILD_GUI=OFF -DCMAKE_INSTALL_PREFIX=/path/to/COPASI/COPASI-Build-217/build -DCOPASI_DEPENDENCY_DIR=/path/to/COPASI/copasi-dependencies-4.26.213/ bin ..
making install
```

Step 9 Run the following commands to add the environment variables:
```bash
export PATH=/path/to/COPASI/COPASI-Build-217/build/bin:$PATH
```

---End

### 4.27.7 Running and Verifying COPASI

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to go to the test case folder provided by the software:
Step 3 Run the following command:

```bash
CopasiSE --nologo EventTest10.cps
```

After the execution is complete, the `EventTest10.1.txt` file is generated. Some content in the file is as follows:

<table>
<thead>
<tr>
<th>Time</th>
<th>[dClk-mRNA]</th>
<th>[PDP1_n]</th>
<th>[VRI_n]</th>
<th>[dCLK_c]</th>
<th>[dCLK_n]</th>
<th>[PER/TIM]</th>
<th>[PER/TIM-dCLK_n]</th>
<th>[vri-mRNA]</th>
<th>[VRI_c]</th>
<th>[pdp1-mRNA]</th>
<th>[PDP1_c]</th>
<th>[per/tim-mRNA]</th>
<th>Values[T2]</th>
<th>Values[periodPDP1_n]</th>
<th>Values[peaktimePDP1_n]</th>
<th>Values[periodPDP1_n2]</th>
<th>Values[entryn]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>1.060865</td>
<td>1.292758</td>
<td>1.14419</td>
<td>1.09696</td>
<td>4.367713e-07</td>
<td>0.2857835</td>
<td>1.473923</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0.4</td>
<td>1.068151</td>
<td>1.648851</td>
<td>1.34179</td>
<td>1.183658</td>
<td>2.294009e-07</td>
<td>0.5653802</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0.89928</td>
<td>0.7703788</td>
<td>1.552953</td>
<td>0.9366577</td>
<td>1.473933</td>
<td>0.9741363</td>
<td>289.439 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1.061505</td>
<td>0.6734762</td>
<td>1.660096</td>
<td>0.9055907</td>
<td>1.679842</td>
<td>0.961366</td>
<td>289.657 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1.260929</td>
<td>0.5871731</td>
<td>1.691519</td>
<td>0.8749563</td>
<td>1.865798</td>
<td>0.9479062</td>
<td>289.8732 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1.467994</td>
<td>0.5105355</td>
<td>1.665151</td>
<td>0.8447758</td>
<td>2.032401</td>
<td>0.9346702</td>
<td>290.087 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1.812153</td>
<td>0.8373431</td>
<td>3.537083</td>
<td>1.986414</td>
<td>1.257421</td>
<td>7.963505e-08</td>
<td>1.631489</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2.060865</td>
<td>0.4426853</td>
<td>1.598318</td>
<td>0.8150713</td>
<td>2.180288</td>
<td>0.9213625</td>
<td>290.2979</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4.147933</td>
<td>0.772229</td>
<td>4.084265</td>
<td>2.041389</td>
<td>1.217347</td>
<td>6.629547e-08</td>
<td>1.881666</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>10.716273</td>
<td>0.382799</td>
<td>1.504186</td>
<td>0.7858652</td>
<td>2.310128</td>
<td>0.9079909</td>
<td>290.5051</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>11.176273</td>
<td>0.3301069</td>
<td>1.393124</td>
<td>0.757182</td>
<td>2.422625</td>
<td>0.8945639</td>
<td>290.7083 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>11.1860603</td>
<td>0.6606023</td>
<td>5.215126</td>
<td>2.018753</td>
<td>1.104975</td>
<td>4.723108e-08</td>
<td>2.392337</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

---End

4.27.8 Troubleshooting

Problem 1: An Error Is Reported After the make install Progress Reaches 100%

Symptom

After the `make install` progress reaches 100%, an error message “make[2]: *** No rule to make target `/path/to/COPASI/copasi-dependencies-master/bin/lib/libexpat.a`, needed by `copasi/CopasiSE/CopasiSE'. Stop." is displayed.

Possible Causes

- The `libexpat.a` file does not exist in the corresponding directory.
- When the script is used to install the dependency, the file is generated in the `lib64` directory at the same level.

Procedure

Step 1 Go to the `copasi-dependencies-4.26.213` directory.

```bash
cd /path/to/COPASI/copasi-dependencies-4.26.213
```
Step 2 Run the following command to copy the files in the lib64 directory to the lib directory:

```
cp -r ./bin/lib64* ./bin/lib
```

---End

4.27.9 More Information

For more resources, visit the COPASI official website:

http://copasi.org/

4.28 FASTA 36.3.8 Porting Guide (CentOS 7.6)

4.28.1 Introduction

The FASTA programs are a comprehensive set of similarity searching and alignment programs for searching protein and DNA sequence databases. Like the BLAST programs blastp and blastn, the fasta program itself uses a rapid heuristic strategy for finding similar regions in protein and DNA sequences. But in addition to heuristic similarity searching, the FASTA package provides programs for rigorous local (ssearch) and global (ggsearch) similarity searching, as well as a program for finding non-overlapping sequence similarities (lalign).

For more information about VarScan, visit the FASTA.

Programming language: C

Brief description: A comprehensive set of similarity searching and alignment programs for searching protein and DNA sequence databases.

Open-source protocol: Apache2.0

Recommended Version

fasta36-36.3.8

4.28.2 Environment Requirements

Hardware Requirements

Table 4-139 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
Software Requirements

Table 4-140 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>FASTA</td>
<td>fasta36-36.3.8</td>
<td><a href="https://github.com/wrpearson/fasta36/archive/v36.3.8h_04-May-2020.tar.gz">https://github.com/wrpearson/fasta36/archive/v36.3.8h_04-May-2020.tar.gz</a></td>
</tr>
<tr>
<td>sse2neon</td>
<td>master</td>
<td><a href="https://github.com/DLTcollab/sse2neon.git">https://github.com/DLTcollab/sse2neon.git</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-141 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.28.3 Planning the Path for Software Porting

Table 4-142 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see Planning Data for Installation in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Usage</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------------</td>
<td>----------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/FASTA</td>
<td>Installation path of FASTA.</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

### 4.28.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.

**Configuration Process**

**Table 4-143 Configuration process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

### 4.28.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the FASTA installation package `fasta36-36.3.8h_04-May-2020.tar.gz`.

Download address: [https://github.com/wrpearson/fasta36/archive/v36.3.8h_04-May-2020.tar.gz](https://github.com/wrpearson/fasta36/archive/v36.3.8h_04-May-2020.tar.gz)

**Step 2** Use the SFTP tool to upload the SU2 installation package to the `/path/to/FASTA` directory on the server.

----End

### 4.28.6 Compiling and Installing FASTA

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.
Step 2  Decompress the installation package.

```
tar -xvf fasta36-36.3.8h_04-May-2020.tar.gz
unzip sse2neon-master.zip
```

Step 3  Switch to the folder generated after the package is decompressed.

```
cp sse2neon-master/sse2neon.h /path/to/FASTA.fasta36-36.3.8h_04-May-2020/src/
cd /path/to/FASTA.fasta36-36.3.8h_04-May-2020/make
```

Step 4  Run the following command to modify the file content:

```
vim Makefile.linux64_sse2
```

1. Press i to edit the `Makefile.linux64_sse2` file and add the following content:

```
13 SHELL=/bin/bash
14 CC = gcc -g -O2 -march=armv8.2-a
16 LIB_DB=
```

2. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 5  Run the following command to switch to the src directory:

```
cd ../src
```

Step 6  Modify the content of the `smith_waterman_sse2.c` file.

1. `vim smith_waterman_sse2.c`

2. Press i to enter the insert mode and modify the `smith_waterman_sse2.c` file as follows:

```
23 #include "sse2neon.h"
24 #endif
```

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 7  Modify the `global_sse2.c` file.

1. Run the `vim` command to open the `global_sse2.c` file.

```
 vim global_sse2.c
```

2. Press i to enter the insert mode and modify the `global_sse2.c` file as follows:

```
21 #else
22 #include "sse2neon.h"
23 #endif
```

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 8  Modify the `glocal_sse2.c` file.

1. Run the `vim` command to open the `glocal_sse2.c` file.

```
 vim glocal_sse2.c
```

2. Press i to enter the insert mode and modify the `glocal_sse2.c` file as follows:

```
21 #else
22 #include "sse2neon.h"
23 #endif
```

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 9  Run the following commands to perform compilation and installation:

```
make -f ../make/Makefile.linux64_sse2
```
Step 10 Run the following command to switch to the data directory:

```
cd /path/to/FASTA/fasta36-36.3.8h_04-May-2020/data
```

----End

4.28.7 Running and Verifying FASTA

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to run fasta36:

```
export PATH=/path/to/FASTA/fasta36-36.3.8h_04-May-2020/bin:$PATH
mpirun --allow-run-as-root -np 96 ../bin/fasta36 -q ../seq/titin_hum.seq
../seq/prot_test.lseg
```

View the total scan time 51.260s, as shown in the following figure:

```
82027 residues in 3 query sequences
1074 residues in 12 library sequences
Tcmltib [36.3.8h May, 2020] (128 proc in memory [06])
Total Scan time: 51.260 Total Display time: 0.000
Function used was FASTA [36.3.8h May, 2020]
```

----End

4.28.8 More Information

Official FASTA website:

http://www.mantevo.org/

4.29 gmap 2015.9.21 Porting Guide (CentOS 7.6)

4.29.1 Introduction

GMAP is a genomic mapping and alignment program for mRNA and EST sequences. GMAP was first used to map EST/cDNA sequences to the reference genome and can be used to annotate the genome structure.

Recommended Software Version

```
gmap-2015-09-21
```

4.29.2 Environment Requirements

Hardware Requirements

Table 4-144 lists the hardware requirements.
Table 4-144 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table Software Requirements lists the software requirements.

Table 4-145 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test case 1</td>
<td>human_g1k_v37.fasta</td>
<td><a href="http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz">http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-146 lists the OS requirements.

Table 4-146 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.29.3 Planning the Paths for Software Porting

This section describes the software installation paths involved in the GMAP software porting.
Table 4-147 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see <strong>Planning Data for Installation</strong> in .</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/GMAP</td>
<td>Installation path of GMAP.</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/TESTCASE</td>
<td>Installation path of human_g1k_v37.fasta.</td>
<td></td>
</tr>
</tbody>
</table>

### 4.29.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.

**Configuration Process**

Table 4-148 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in <strong>HPC Solution Basic Environment Setup Guide</strong>.</td>
</tr>
</tbody>
</table>

### 4.29.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the GMAP installation package **gmap-gsnap-2015-09-21.tar.gz**.


**Step 2** Download the **human_g1k_v37.fasta** file.
Download address: http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz.

**Step 3** Use SFTP to upload the GMAP installation package to the /path/to/GMAP directory on the server.

**Step 4** Use SFTP to upload the human_g1k_v37.fasta test case to the /path/to/TESTCASE directory on the server.

---End

### 4.29.6 Compiling and Installing GMAP

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Decompress the installation package.

```
tar -zxvf gmap-gsnap-2015-09-21.tar.gz
```

**Step 3** Switch to the folder generated after the package is decompressed.

```
cd /path/to/GMAP/gmap-2015-09-21
```

**Step 4** Run the following command to perform configuration:

```
./configure --prefix=/path/to/GMAP/Install --build=arm-linux
```

**Step 5** Run the following commands to compile and install SU2:

```
make -j 32
make install
```

**Step 6** Run the following commands to add the environment variables:

```
export PATH=/path/to/GMAP/Install/bin:$PATH
```

---End

### 4.29.7 Running and Verifying GMAP

**Procedure**

**Step 1** Run the following command to view the GMAP version:

```
gmap --verion
```

The installation is successful if the following information is displayed:

![GMAP Version Information](image-url)

---
Step 2  Run the following command to decompress the test case:

```bash
cd path/to/TESTCASE
gzip -d human_g1k_v37.fasta.gz
```

Step 3  Run the following command to create a reference sequence index (-D: index path; -d: index file):

```bash
mkdir human_index_home

gmap_build -D /path/to/TESTCASE/human_index_home -d human_g1k_v37
human_g1k_v37.fasta
```

Step 4  Run a series of commands to compare the files. (The `-f` parameter is used to output files in the gff3_gene format.)

```bash
{ time -p gmap -t 96 -D /path/to/TESTCASE/human_index_home -d human_g1k_v37 -f gff3_gene human_g1k_v37.fasta; } 2>&1 |tee -a mapping_human.gff3
```

If the `mapping_human.gff3` file is generated, the execution is successful.

4.30 VarScan 2.4.2 Porting Guide (CentOS 7.6)

4.30.1 Introduction

VarScan is a software tool developed at the Genome Institute at Washington University to detect variants in NGS data. VarScan employs a robust heuristic/statistic approach to call variants that meet desired thresholds for read depth, base quality, variant allele frequency, and statistical significance.

For more information about VarScan, visit the official VarScan website.

Programming language: java

Brief description: A software tool used to detect variants in NGS data.

Recommended Version

VarScan 2.4.2
4.30.2 Environment Requirements

Hardware Requirements

Table 4-149 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-150 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>VarScan</td>
<td>2.4.4</td>
<td><a href="https://codeload.github.com/dkoboldt/varscan/tar.gz/2.4.2">https://codeload.github.com/dkoboldt/varscan/tar.gz/2.4.2</a></td>
</tr>
<tr>
<td>Samtools</td>
<td>1.10</td>
<td><a href="https://codeload.github.com/samtools/samtools/tar.gz/1.10">https://codeload.github.com/samtools/samtools/tar.gz/1.10</a></td>
</tr>
<tr>
<td></td>
<td>bwa.sam</td>
<td>For details about the BWA result file, see BWA 0.7.17 Porting Guide (CentOS 7.6).</td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-151 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.30.3 Planning the Path for Software Porting

This section describes the software installation paths involved in the VarScan software porting.
### Table 4-152 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see “Planning the Installation Paths” in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/SAMTOOLS</td>
<td>Installation path of samtools.</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/VARSCAN</td>
<td>Installation path of VarScan.</td>
<td></td>
</tr>
</tbody>
</table>

### 4.30.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configur e the basic environ ment.</td>
<td>For details, see “Setting Up the Single-Node System Environment” in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

### 4.30.5 Obtaining the Source Code

#### Procedure

**Step 1** Download the **samtools-1.10.tar.gz** package.

URL: [https://codeload.github.com/samtools/samtools/tar.gz/1.10](https://codeload.github.com/samtools/samtools/tar.gz/1.10).

**Step 2** Download the VarScan installation package **varscan-2.4.2.tar.gz**.

URL: [https://codeload.github.com/dkoboldt/varscan/tar.gz/2.4.2](https://codeload.github.com/dkoboldt/varscan/tar.gz/2.4.2).
4.30.6 Compiling and Installing VarScan

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies:

```
yum install ncurses-devel.aarch64 bzip2-devel.aarch64 xz-devel.aarch64 java-1.8.0-openjdk-devel.aarch64 -y
```

Step 3 Run the following command to switch to the SAMTOOLS directory:

```
cd /path/to/SAMTOOLS
```

Step 4 Run the following commands to decompress the samtools package:

```
tar -xvf samtools-1.10.tar.gz
```

Step 5 Switch to the folder generated after the package is decompressed.

```
cd samtools-1.10
```

Step 6 Run the following command to perform configuration:

```
./configure --prefix=/path/to/SAMTOOLS/samtools-1.10-build
```

Step 7 Run the following command to perform compilation and installation:

```
make -j16
make install
```

Step 8 Run the following command to add samtools the environment variables:

```
export PATH=/path/to/SAMTOOLS/samtools-1.10-build/bin:$PATH
```

Step 9 Run the following command to switch to the VARSCAN directory:

```
cd /path/to/VARSCAN
```

Step 10 Run the following command to decompress varscan-2.4.2.tar.gz:

```
tar -xvf varscan-2.4.2.tar.gz
```

Step 11 Switch to the folder generated after the package is decompressed.

```
cd varscan-2.4.2
```

Step 12 Run the following commands to test the VarScan environment:

```
java -jar VarScan.v2.4.2.jar
```

If the environment is normal, the following information is displayed:
4.30.7 Running and Verifying VarScan

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to copy the test case files and go to the test directory:

```
cp hg19.fa /path/to/VARSCAN/varscan-2.4.2

cp bwa.sam /path/to/VARSCAN/varscan-2.4.2

cd /path/to/VARSCAN/varscan-2.4.2
```

Step 3 Run the following commands to use samtools to convert the format and sort the data:

```
samtools view -bS bwa.sam > bwa.bam

samtools sort -@96 -o bwa.sort.bam bwa.bam
```

Step 4 Run the following command to use samtools to process the case files:

```
samtools mpileup -f hg19.fa bwa.sort.bam > myData.pileup
```

Step 5 Run the following command to run the test case:

```
{ time java -jar VarScan.v2.4.2.jar pileup2snp myData.pileup > VarScan.vcf ; } 2>&1 |tee varscan.log
```

After the execution is complete, the VarScan.vcf and varscan.log files are generated in the current directory. VarScan.vcf is the case comparison result file, and varscan.log is the run log file.

Check the value of real (in seconds) in the varscan.log log file. A smaller value indicates better performance.

Figure 4-14 is an example of the output result.
4.30.8 More Information

Official VarScan website:

http://dkoboldt.github.io/varscan/

4.31 Minimap2 Porting Guide (CentOS 7.6)

4.31.1 Introduction

Minimap2 is a versatile sequence alignment program that aligns DNA or mRNA sequences against a large reference database.

Programming language: C/C++

Brief description: versatile sequence alignment program

Open-source license: MIT

Recommended Version

Obtain the latest version using Git.

4.31.2 Environment Requirements

Hardware Requirements

Table 4-154 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
Software Requirements

*Table Software requirements* lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>zlib</td>
<td>zlib-1.2.7-18</td>
<td><a href="https://github.com/madler/zlib">https://github.com/madler/zlib</a></td>
</tr>
</tbody>
</table>

OS Requirements

*Table 4-156* lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.31.3 Planning the Path for Software Porting

This section describes the software installation paths involved in the Minimap2 software porting.

*Table 4-157* Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>
### 4.31.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

#### Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Configure the basic environment.</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
</tbody>
</table>

### 4.31.5 Obtaining the Source Code

#### Procedure

**Step 1**  
Download the mimimap2 installation package `minimap2-master.zip`.  
**URL:** [https://github.com/lh3/minimap2/archive/master.zip](https://github.com/lh3/minimap2/archive/master.zip)

**Step 2**  
Use SFTP to upload the mimimap2 installation package to the `/path/to/ MINIMAP2` directory on the server.

----End

### 4.31.6 Compiling and Installing Minimap2

#### Procedure

**Step 1** Use PuTTY to log in to the server as the `root` user.

**Step 2** Configure the compilation environment and install the `wget` tool.
yum install wget –y

**Step 3** Install the Minimap2 dependency library and tool.

yum install –y git gcc-c++ zlib -y

**Step 4** Decompress the installation package.

cd /path/to/MINIMAP2

unzip minimap2-master.zip

**Step 5** Compile and install minimap2. Kunpeng is compatible with the Armv8 64-bit instruction set. Therefore, the compilation method is as follows:

cd /path/to/MINIMAP2/minimap2-master

make arm_neon=1 aarch64=1

**Step 6** Run and verify minimap2. Check the version of minimap2.

cd /path/to/MINIMAP2/minimap2-master

./minimap2 --v

Minimap2 is compiled correctly if the following information is displayed:

2.17-r974-dirty

-----End

### 4.31.7 Running and Verifying Minimap2

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to switch to the working directory:

cd /path/to/MINIMAP2/minimap2-master

**Step 3** Run the following command to run the case:

./minimap2 -a test/MT-human.fa test/MT-orang.fa > test.sam

If the command is successfully executed, the following information is displayed:

```
[M::mm_idx_gen::0.002*2.34] collected minimizers
[M::mm_idx_gen::0.006*2.70] sorted minimizers
[M::main::0.006*2.69] loaded/built the index for 1 target sequence(s)
[M::mm_mapopt_update::0.007*2.51] mid_occ = 2
[M::mm_idx_stat::0.007*2.40] kmer size: 15; skip: 10; is_hpc: 0; #seq: 1
[M::mm_idx_stat::0.007*2.40] distinct minimizers: 3111 (100.00% are singletons); average occurrences: 1.000; average spacing: 5.326; total length: 16569
[M::worker_pipeline::0.032*1.32] mapped 1 sequences
[M::main] Version: 2.17-r974-dirty
[M::main] CMD: ./minimap2 -a test/MT-human.fa test/MT-orang.fa
[M::main] Real time: 0.035 sec; CPU: 0.045 sec; Peak RSS: 0.005 GB
```

-----End
4.32 bedtools 2.29.2 Porting Guide (CentOS 7.6)

4.32.1 Introduction

bedtools is a powerful set of utilities for genomics analysis tasks. For example, bedtools allows one to intersect, merge, count, complement, and shuffle genomic intervals from multiple files in widely-used genomic file formats such as BAM, BED, GFF/GTF, VCF. While each individual tool is designed to do a relatively simple task (e.g., intersect two interval files), quite sophisticated analyses can be conducted by combining multiple bedtools operations on the Unix command line.

For more information about bedtools, visit the official bedtools website.

Programming language: C/C++

Brief description: a powerful set of utilities for genomics analysis tasks

Open-source license: GNU

Recommended Version

bedtools-2.29.2

4.32.2 Environment Requirements

Hardware Requirements

Table 4-159 lists the hardware requirements.

Table 4-159 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-160 lists the software requirements.

Table 4-160 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>bedtools</td>
<td>2.29.2</td>
<td><a href="https://github.com/arq5x/bedtools2/releases/download/v2.29.2/bedtools-2.29.2.tar.gz">https://github.com/arq5x/bedtools2/releases/download/v2.29.2/bedtools-2.29.2.tar.gz</a></td>
</tr>
<tr>
<td>Test case</td>
<td>aluY.chr1.bed</td>
<td>Test case provided by the software</td>
</tr>
<tr>
<td></td>
<td>gerp.chr1.bed</td>
<td>Test case provided by the software</td>
</tr>
</tbody>
</table>
OS Requirements

Table 4-161 lists the OS requirements.

Table 4-161 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.32.3 Planning the Paths for Software Porting

Table 4-162 describes the software installation paths involved in the bedtools software porting.

Table 4-162 Paths for software porting

<table>
<thead>
<tr>
<th>N o.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the <a href="https://www.centos.org/download/">HPC Solution Basic Environment Setup Guide</a>. The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/BEDTOOLS</td>
<td>Installation path of BCFtools.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>/path/to/TESTCASE</td>
<td>Path for storing test cases.</td>
<td></td>
</tr>
</tbody>
</table>

4.32.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.
4.32.5 Obtaining the Source Code

Procedure

Step 1 Download the `bedtools-2.29.2.tar.gz` installation package.
URL: https://github.com/arq5x/bedtools2/releases/download/v2.29.2/bedtools-2.29.2.tar.gz

Step 2 Use the SFTP tool to upload the bedtools installation package to the `/path/to/BEDTOOLS` directory on the server.

4.32.6 Compiling and Installing BCFtools

Step 1 Install related dependencies.

```
yum install zlib-devel bzip2-devel xz-devel -y
```

Step 2 Run the following command to decompress the installation package:

```
tar -zxvf bedtools-2.29.2.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
cd bedtools2
```

Step 4 Run the following command to perform installation and compilation.

```
make
```

Step 5 Run the following command to add the environment variable:

```
export PATH=/path/to/BEDTOOLS/bedtools2/bin:$PATH
```

----End
4.32.7 Running and Verifying bedtools

Procedure

Step 1  Run the following commands to copy the example in the data directory to the /path/to/TESTCASE directory:

```bash
cd /path/to/BEDTOOLS/bedtools2/data
cp gerp.chr1.bed.gz aluY.chr1.bed.gz /path/to/TESTCASE
```

Step 2  Run the following commands to decompress the test case packages:

```bash
cd /path/to/TESTCASE
gzip -d gerp.chr1.bed.gz
gzip -d aluY.chr1.bed.gz
```

Step 3  Run the following command to generate the overlap of two intersecting files:

```bash
bedtools intersect -a aluY.chr1.bed -b gerp.chr1.bed
```

If information similar to the following is displayed, the command is successfully run.

```
chr1 8837562 8837599 AluY 2462 +
chr1 9986239 9986242 AluY 2473 +
chr1 12065383 12065381 AluYf4 2547 +
chr1 12687654 12687666 AluY 2376 -
chr1 16236644 16236646 AluY 2711 +
chr1 38030286 38030281 AluY 2396 -
chr1 38054001 38054002 AluY 2363 -
chr1 52927847 52927849 AluYk4 2636 -
chr1 54333040 54333042 AluY 2636 -
chr1 64044117 64044122 AluY 2579 -
chr1 68293406 68293407 AluY 2440 -
chr1 72383470 72383481 AluYc3 2576 -
chr1 75772409 75772410 AluY 2731 -
chr1 102695959 102695965 AluYb8 2927 +
chr1 106555263 106555264 AluY 2578 -
chr1 107555937 107555938 AluY 2445 +
chr1 108603179 108603175 AluY 2695 +
chr1 14557317 14557317 AluYf5 2742 +
chr1 159999185 159999187 AluY 2151 -
chr1 188601987 188601901 AluY 2721 -
chr1 199170536 199170537 AluY 2408 +
chr1 209246451 209246456 AluY 2647 -
chr1 211477823 211477825 AluYg6 990 +
chr1 226371087 226371914 AluY 2420 +
chr1 236031459 236031463 AluYb8 672 -
chr1 237494801 237494803 AluY 2408 -
```

----End

4.33 BEAST2 2.6.3 Porting Guide (CentOS 7.6)

4.33.1 Introduction

BEAST 2 is a cross-platform program for Bayesian phylogenetic analysis of molecular sequences. It estimates rooted, time-measured phylogenies using strict
or relaxed molecular clock models. It can be used as a method of reconstructing
phylogenies but is also a framework for testing evolutionary hypotheses without
conditioning on a single tree topology. BEAST 2 uses Markov chain Monte Carlo
(MCMC) to average over tree space, so that each tree is weighted proportional to
its posterior probability. BEAST 2 includes a graphical user-interface for setting up
standard analyses and a suite of programs for analysing the results.

For more information, visit the **BEAST 2 official website**.
Programming language: Java
Open-source license: GPL

**Recommended Version**

BEAST V2.6.3

**4.33.2 Environment Requirements**

**Hardware Requirements**

Table 4-164 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

**Software Requirements**

Table 4-165 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEAST 2</td>
<td>V2.6.3</td>
<td><a href="https://github.com/CompEvol/beast2/releases/download/v2.6.3/BEAST.v2.6.3.Linux.tgz">https://github.com/CompEvol/beast2/releases/download/v2.6.3/BEAST.v2.6.3.Linux.tgz</a></td>
</tr>
<tr>
<td>Test case</td>
<td>Primates.nex</td>
<td>Test case provided by the software</td>
</tr>
</tbody>
</table>

**OS Requirements**

Table 4-166 lists the OS requirements.
<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.33.3 Planning the Paths for Software Porting

This section describes the software installation paths involved in the BEAST 2 software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/BEAST</td>
<td>Installation path of BEAST 2.</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

### 4.33.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.
### Configuration Process

#### Table 4-168 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
</tbody>
</table>

### 4.33.5 Obtaining the Source Code

**Procedure**

- **Step 1** Download the **BEAST.v2.6.3.Linux.tgz** installation package.
  
  URL: [https://github.com/CompEvol/beast2/releases/download/v2.6.3/BEAST.v2.6.3.Linux.tgz](https://github.com/CompEvol/beast2/releases/download/v2.6.3/BEAST.v2.6.3.Linux.tgz)

- **Step 2** Use SFTP to upload the BEAST 2 installation package to the */path/to/BEAST* directory on the server.

----End

### 4.33.6 Compiling and Installing BEAST 2

**Procedure**

- **Step 1** Use PuTTY to log in to the server as the **root** user.

- **Step 2** Run the following command to install the Java package:
  
  ```
  yum install java-1.8.0-openjdk.aarch64
  ```

- **Step 3** Run the following command to decompress the installation package:
  
  ```
  tar -zxvf BEAST.v2.6.3.Linux.tgz
  ```

- **Step 4** Run the following command to add the environment variable:
  
  ```
  export PATH=/path/to/BEAST/beast/bin:SPATH
  ```

----End

### 4.33.7 Running and Verifying BEAST 2

The **Primates.nex** data set is provided in the **nexus** directory of the BEAST 2 installation package. You can use BEAUti to configure parameters and generate XML files. After the generated XML files are imported, run the **beast** command to generate the log file (*log*), tree file (*trees*), and XML status file (*state*).
Procedure

**Step 1** Run the following command to switch to the `nexus` directory:
```
cd /path/to/BEAST/beast/examples/nexus
```

**Step 2** Run the following command. The **BEAUti 2: Standard** window is displayed, as shown in Figure 4-15.
```
beauti
```

![Figure 4-15 BEAUti 2: Standard window](image)

**Step 3** Import a sequence. Choose **File > Import Alignment**. The **Open** window is displayed. Select the `Primates.nex` sequence file in the `nexus` directory, as shown in Figure 4-16.

![Figure 4-16 Primates.nex sequence file](image)

**Step 4** Link or unlink partition models. Click **Open** and the following dialog box is displayed. Generally, the imported gene sequence information is aligned. However,
in some cases, partitioned sequence exists. If joint analysis is required, the information needs to be linked. If joint analysis is not required, the information needs to be unlinked.

**Step 5** Set the substitution model. For the nucleotide substitution model, you can select GTR, HKY, JC69, or TN93. **Gamma Category Count** can be set to a value ranging from 4 to 8. HKY and 4 are selected here as an example. Other parameters such as **Substitution Rate, Shape** (the shape parameter of gamma distribution), and **Frequencies** are estimated in the analysis.

**Step 6** Set molecular clock.

Click the **Clock Model** tab. Parameters include Strict Clock, Relaxed Clock Exponential, and Relaxed Clock Log Normal. If Strict Clock is selected, the rate difference between molecules in the model is not considered.
Step 7  Set verification information.
1. Click Priors and set Tree.t:firsthalf and Tree.t:secondhalf to Yule Model. Retain the default settings for other parameters.

2. Click +Add Prior at the bottom of the Priors area. In the window that is displayed, select Tree.t:secondhalf. In Taxon set editor, define a name, for example, fossil information. Select two species (for example, Homo_sapiens and Pan) whose divergence time needs to be calibrated and move them to the right border.
Select **Normal** in the monophyletic drop-down list to verify the distribution information. Set the divergence time. The divergence of Homo sapiens and Pan is between 5 and 7 My. Therefore, set the center point of the normal distribution to 6 My and the standard deviation to 0.5 My, and select **monophyletic**.

**Step 8** Click **MCMC** to set the MCMC options.
Chain Length: 10,000,000 (by default)
tracelog: The chain is sampled once every 1000 times.
screenlog: The log is displayed on the screen every 1000 times.
treelog.t:tree: The topology is printed every 1000 times.

**Step 9** Save the XML file.

Choose **File > Save In** and define an output file name to generate an XML file.

```
step 10
```

Run the following command to perform the beast test:
```
beast -threads 128 test01.xml
```

Check the value of **Total calculation time** in the **Primates.log** file. The unit is second. A smaller value indicates better performance. The following is an example of the output result.
4.33.8 More Information

Obtain BEAST related documents at http://beast.community/

4.34 bcftools 1.10.2 Porting Guide (CentOS 7.6)

4.34.1 Introduction

BCFtools is a tool provided by Samtools developers to manipulate files in the variant call format (VCF). It can process VCF files as well as their binary counterparts BCF files. BCFtools consists of many commands for analyzing genotypic variants, manipulating VCF files, and creating indexes for VCF/BCF files.

Programming language: C

Open-source license: MIT/Expat

Recommended Version

bcftools-1.10.2

4.34.2 Environment Requirements

Hardware Requirements

Table 4-169 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-170 lists the software requirements.
Table 4-170 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>bcftools</td>
<td>1.10.2</td>
<td><a href="https://github.com/samtools/bcftools/releases/download/1.10.2/bcftools-1.10.2.tar.bz2">https://github.com/samtools/bcftools/releases/download/1.10.2/bcftools-1.10.2.tar.bz2</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-171 lists the OS requirements.

Table 4-171 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.34.3 Planning the Paths for Software Porting

Table 4-172 describes the software installation paths involved in the BCFtools software porting.

Table 4-172 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/BCFTOOLS</td>
<td>Installation path of BCFtools.</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>
4.34.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.

Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

4.34.5 Obtaining the Source Code

Procedure

Step 1 Download the BCFtools-1.10.2.tar.bz2 installation package.

URL: https://github.com/Samtools/BCFtools/releases/download/1.10.2/BCFtools-1.10.2.tar.bz2

Step 2 Use the SFTP tool to upload the BCFtools installation package to the /path/to/BCFTOOLS directory on the server.

----End

4.34.6 Compiling and Installing BCFtools

Procedure

Step 1 Install related dependencies.

yum install zlib-devel bzip2 bzip2-devel xz-devel libcurl-devel -y

Step 2 Run the following commands to decompress the installation package:

tar jxvf bcf tools-1.10.2.tar.bz2

Step 3 Run the following command to switch to the directory generated after decompression:

cd bcf tools-1.10.2

Step 4 Run the following command to perform configuration:

./configure --prefix=/path/to/BCFTOOLS/Install CFLAGS="-O3 -march=armv8.2-a"
Step 5  Run the following commands to compile and install BCFtools:

```
make -j 32
make install
```

Step 6  Run the following command to add the environment variable:

```
export PATH=/path/to/BCFTOOLS/Install/bin:$PATH
```

----End

### 4.34.7 Running and Verifying BCFtools

#### Procedure

**Step 1**  View the BCFtools version.

Run the following command to view the version, as shown in Figure 4-17.

```
bcftools --version
```

**Figure 4-17 Version information**

```
[root@ARMX bcftools-1.10.2] # bcftools
Program: bcftools (Tools for variant calling and manipulating VCFs and BCFs)
Version: 1.10.2 (using htslib 1.10.2)
```

**Step 2**  Test data.

Run the following commands to test the example in the test directory:

```
cd bcftools-1.10.2
make test
```

If the following information is displayed, the BCFtools test is successful, as shown in Figure 4-18.

**Figure 4-18 Successful BCFtools test**

```
Number of tests:
  total .. 1141
  passed .. 1141
  failed .. 0
```

----End

### 4.35 Picard 2.23.3 Porting Guide (CentOS 7.6)

#### 4.35.1 Introduction

Picard is a set of command line tools for manipulating high-throughput sequencing (HTS) data and formats such as SAM/BAM/CRAM and VCF.
For more information about Picard, visit the official Picard website.
Programming language: java
Brief description: toolkit used to process HTS data and formats
Open-source license: MIT

**Recommended Version**

Picard 2.23.3

**4.35.2 Environment Requirements**

**Hardware Requirements**

Table 4-174 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

**Software Requirements**

Table 4-175 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picard</td>
<td>2.23.3</td>
<td><a href="https://github.com/broadinstitute/picard/releases/download/2.23.3/picard.jar">https://github.com/broadinstitute/picard/releases/download/2.23.3/picard.jar</a></td>
</tr>
<tr>
<td>Samtools</td>
<td>0.1.9</td>
<td><a href="https://codeload.github.com/samtools/samtools/tar.gz/0.1.9">https://codeload.github.com/samtools/samtools/tar.gz/0.1.9</a></td>
</tr>
<tr>
<td>B17NC_R1.fq.gz</td>
<td></td>
<td><a href="https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz">https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz</a></td>
</tr>
<tr>
<td>B17NC_R2.fq.gz</td>
<td></td>
<td><a href="https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R2.fastq.gz">https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R2.fastq.gz</a></td>
</tr>
</tbody>
</table>
OS Requirements

Table 4-176 lists the OS requirements.

Table 4-176 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.35.3 Planning the Paths for Software Porting

Table 4-177 describes the software installation paths involved in the Picard software porting.

Table 4-177 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the <a href="#">HPC Solution Basic Environment Setup Guide</a>. The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/PICARD</td>
<td>Installation path of Picard.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>/path/to/SAMTOOLS</td>
<td>Installation path of Samtools.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/TESTCASE</td>
<td>Installation path of test cases.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/BWA</td>
<td>Installation path of BWA.</td>
<td>For details, see the BWA 0.7.17 Porting Guide (CentOS 7.6).</td>
</tr>
</tbody>
</table>

4.35.4 Configuring the Compilation Environment

Prerequisites

The installation packages are uploaded to the server using an SFTP tool.
Configuration Process

Table 4-178 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

4.35.5 Obtaining the Source Code

Procedure

Step 1  Download the samtools-0.1.9.tar.bz2 package.

URL: https://codeload.github.com/samtools/samtools/tar.gz/0.1.9

Step 2  Download the piccard.jar installation package.

URL: https://github.com/broadinstitute/picard/releases/download/2.23.3/picard.jar

Step 3  Use the SFTP tool to upload the Samtools package to the /path/to/SAMTOOLS directory on the server, and upload the Picard installation package to the /path/to/PICARD directory on the server.

----End

4.35.6 Running and Installing Picard

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to install dependencies:

```
yum install ncurses-devel.aarch64 bzip2-devel.aarch64 xz-devel.aarch64 java-1.8.0-openjdk-devel.aarch64 -y
```

Step 3  Run the following command to switch to the Samtools directory:

```
cd /path/to/SAMTOOLS
```

Step 4  Run the following command to decompress the Samtools package:

```
tar -xvf samtools-0.1.9.tar.gz
```

Step 5  Run the following command to switch to the directory generated after decompression:

```
cd samtools-0.1.9
```

Step 6  Run the following command to obtain the absolute path of GNU:
which gcc

**Step 7** Run the following command to modify the **Makefile** file:

```
vi Makefile
```

Before the modification:
```
CC     = gcc
```

After the modification:
```
CC     = /path/to/GNU/bin/gcc
```

**Step 8** Run the following command to perform compilation:

```
make
```

**Step 9** Run the following command to add the environment variable of Samtools:

```
export PATH=/path/to/SAMTOOLS/samtools-0.1.9:$PATH
```

**Step 10** Run the following command to go to the Picard directory:

```
cd /path/to/PICARD
```

**Step 11** Run the following command to test the Picard environment:

```
java -jar picard.jar
```

The following figure shows that the environment is normal.

---

**4.35.7** Running and Verifying Picard

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following commands to copy the test case files and go to the test directory:

```
 cp B17NC_R1.fq.gz /path/to/TESTCASE
 cp B17NC_R2.fq.gz /path/to/TESTCASE
 cp GCA_000001405.15_GRCh38_full_analysis_set.fna.gz /path/to/TESTCASE
```
cd /path/to/TESTCASE

Step 3 Run the following commands to decompress the case files:

gzip -d B17NC_R1.fastq.gz

gzip -d B17NC_R2.fastq.gz

gzip -d GCA_000001405.15_GRCh38_full_analysis_set.fna.gz

Step 4 Run the following command to rename the
GCA_000001405.15_GRCh38_full_analysis_set.fna.gz file to hs38DH.fasta:

mv GCA_000001405.15_GRCh38_full_analysis_set.fna hs38DH.fasta

Step 5 Run the following commands to add the environment variables of BWA:

export PATH=/path/to/BWA/bwa-0.7.17:$PATH

export PLATFORM="Illumina"

export RG_ID="L1"

export LIB="pt2"

export SAMPLE=B17NC

Step 6 Run the following command to process the test case files using BWA:

bwa index -a bwtsw hg19.fa

bwa mem -t 96 -o B17NC.sam -M -R "@RG\tID:$RG_ID\tPL:$PLATFORM\tSM: $SAMPLE\tCN:$LIB" hg19.fa B17NC_R1.fastq B17NC_R2.fastq

Step 7 Run the following commands to use Samtools to convert the format and sort the
data:

samtools view -bS B17NC.sam > B17NC.bam

samtools sort $SAMPLE.bam $SAMPLE.sort

Step 8 Run the following command to run the test case:

{ time java -jar picard.jar CollectAlignmentSummaryMetrics R=hs38DH.fasta I=B17NC.sort.bam O=output.txt; } 2>&1 |tee picard.log

After the command is run, the output.txt and picard.log files are generated in the
current directory. The output.txt file contains the output and the picard.log file
contains the run logs.

Step 9 Check the value of Elapsed time in the picard.log file. The unit is minute. A
smaller value indicates better performance.

Figure 4-19 shows an example.
### 4.35.8 Troubleshooting

**Problem 1: An error is reported when the Picard command is run.**

**Symptom**

An error message similar to the following is displayed when Picard is run: "Unable to load libgkl_compression.so from native/libgkl_compression.so (/tmp/libgkl_compression8107837509941713683.so: /tmp/libgkl_compression8107837509941713683.so: cannot open shared object file: No such file or directory (Possible cause: can't load AMD 64-bit .so on a AARCH64-bit platform))."

**Possible Cause**

Genomics Kernel Library (GKL) is an acceleration library developed based on the Intel CPU, and needs to be ported to the Kunpeng platform.

**Solution**

Ignore this error if performance comparison is not involved.

### 4.36 SOAPdenovo r241 Porting Guide (CentOS 7.6)

#### 4.36.1 Introduction

SOAPdenovo is a novel short-read assembly method that can build a de novo draft assembly for the human-sized genomes. The software is specially designed to assemble Illumina GA short reads. The new version reduces memory consumption during graph creation, resolves repeats in contig steps, increases the coverage and length of scaffold assembly, and improves gap closure. Therefore, SOAPdenovo is more suitable for large genome assembly.

For more information about SOAPdenovo, visit the [official SOAPdenovo website](#).

Programming language: C/C++

Brief description: software for genome assembly

**Recommended Version**

SOAPdenovo r241
4.36.2 Environment Requirements

Hardware Requirements

Table 4-179 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-180 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAPd enovo</td>
<td>r241</td>
<td><a href="https://codeload.github.com/aquaskyline/SOAPdenovo2/tar.gz/r241">https://codeload.github.com/aquaskyline/SOAPdenovo2/tar.gz/r241</a></td>
</tr>
<tr>
<td>Test case</td>
<td>B17NC_R1.fq.gz</td>
<td><a href="https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz">https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz</a></td>
</tr>
<tr>
<td></td>
<td>B17NC_R2.fq.gz</td>
<td><a href="https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R2.fastq.gz">https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R2.fastq.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-181 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.36.3 Planning the Paths for Software Porting

Table 4-182 describes the software installation paths involved in the SOAPdenovo software porting.
### Table 4-182 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/SOAPDENOVO</td>
<td>Installation path of SOAPdenovo.</td>
<td>The installation paths listed in this table are only examples. Shared paths are recommended. All the paths used in the commands in this document are examples only. Use the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

### 4.36.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.

**Configuration Process**

#### Table 4-183 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting up the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
</tbody>
</table>

### 4.36.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the SOAPdenovo2-r241.tar.gz installation package.

**URL:** https://codeload.github.com/aquaskyline/SOAPdenovo2/tar.gz/r241

**Step 2** Use the SFTP tool to upload the SOAPdenovo installation package to the /path/to/SOAPDENOVO directory on the server.

----End
4.36.6 Compiling and Installing SOAPdenovo

Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following command to go to the SOAPdenovo directory:

```
    cd /path/to/SOAPDENOVO
```

**Step 3** Run the following command to decompress **SOAPdenovo2-r241.tar.gz**:

```
    tar -xvf SOAPdenovo2-r241.tar.gz
```

**Step 4** Run the following command to switch to the directory generated after decompression:

```
    cd SOAPdenovo2-r241
```

**Step 5** Run the following command to modify the file:

```
    vi standardPregraph/hashFunction.c
```

Before the modification:

```c
static uint32_t cpuid ( uint32_t functionInput )
{
    uint32_t eax,
    uint32_t ebx;
    uint32_t ecx;
    uint32_t edx;
    #ifdef __PIC__
    asm ( "pushl %ebx\n|t" /* save %ebx */
                      "cpuid\n|t"
                      "movl %ebx, %[ebx]\n|t" /* save what cpuid just put in %ebx */
                      "popl %eax":"=a" ( eax ), [ebx] "=r" ( ebx ), "=c" ( ecx ), "=d" ( edx ) : "a" ( functionInput )
                      ":"cc" );
    #else
    asm ( "cpuid" : "=a" ( eax ), "=b" ( ebx ), "=c" ( ecx ), "=d" ( edx ) : "a" ( functionInput ) );
    #endif
    return ecx;
}
```

After the modification:

```c
static uint32_t cpuid ( uint32_t functionInput )
{
    uint32_t eax,
    uint32_t ebx;
    uint32_t ecx;
    uint32_t edx;
    #ifdef __aarch64__
    ecx = 0x0;
    #else
    #ifdef __PIC__
    asm ( "pushl %ebx\n|t" /* save %ebx */
                      "cpuid\n|t"
                      "movl %ebx, %[ebx]\n|t" /* save what cpuid just put in %ebx */
                      "popl %eax":"=a" ( eax ), [ebx] "=r" ( ebx ), "=c" ( ecx ), "=d" ( edx ) : "a" ( functionInput )
                      ":"cc" );
    #else
    asm ( "cpuid" : "=a" ( eax ), "=b" ( ebx ), "=c" ( ecx ), "=d" ( edx ) : "a" ( functionInput ) );
    #endif
    #endif
    return ecx;
}
```
Step 6 Run the following command to perform compilation:

```
make
```

Step 7 Run the following command to test the SOAPdenovo environment:

```
./SOAPdenovo-63mer
```

The following figure shows that the environment is normal.

#### End

4.36.7 Running and Verifying SOAPdenovo

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to copy the test case files and go to the test directory:

```
 cp B17NC_R1.fq.gz /path/to/SOAPDENOVO/SOAPdenovo2-r241
 cp B17NC_R2.fq.gz /path/to/SOAPDENOVO/SOAPdenovo2-r241
 cd /path/to/SOAPDENOVO/SOAPdenovo2-r241
```

Step 3 Run the following commands to decompress the case files:

```
gzip -d B17NC_R1.fastq.gz
 gzip -d B17NC_R2.fastq.gz
```

Step 4 Run the following command to clear the cache:

```
echo 3 > /proc/sys/vm/drop_caches
```

Step 5 Run the following command to create the configuration file:

```
vi config_file
```

```bash
max_rd_len=100
[LIB]
avg_ins=200
reverse_seq=0
asm_flags=3
rd_len_cutoff=100
rank=1
q1=B17NC_R1.fastq
q2=B17NC_R2.fastq
```
**Step 6** After the modification is complete, enter `:wq` to save the modification and exit.

**Step 7** Run the following command to run the test case:

```plaintext
{ time -p ./SOAPdenovo-63mer all -s config_file -K 63 -p 96 -R -o graph_prefix; } 2>&1 | tee SOAPdenovo.log
```

Check the value of `real` in the `SOAPdenovo.log` log file. The unit is second. A smaller value indicates better performance.

*Figure 4-20* shows an example.

---

**4.36.8 More Information**

Official SOAPdenovo website:

[https://github.com/aquaskyline/SOAPdenovo2](https://github.com/aquaskyline/SOAPdenovo2)

**4.37 Blastz 2004-12-27 Porting Guide (CentOS 7.6)**

**4.37.1 Introduction**

Blastz is a nucleotide local alignment program developed by Webb Miller's group. It takes FASTA or nib sequences as input and produces output in its own format, lav. It has some notable algorithmic improvements over traditional blast.

For more information about Blastz, visit the [Blastz official website](https://github.com/aquaskyline/SOAPdenovo2).

**Programming language:** C

**Brief description:** a nucleotide local alignment program

**Open-source license:** GPL

**Recommended Version**

Blastz 2004-12-27.
4.37.2 Environment Requirements

Hardware Requirements

Table 4-184 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-185 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blastz</td>
<td>2004-12-27</td>
<td><a href="http://www.bx.psu.edu/miller_lab/dist/blastz-2004-12-27.deprecated.tar.gz">http://www.bx.psu.edu/miller_lab/dist/blastz-2004-12-27.deprecated.tar.gz</a></td>
</tr>
<tr>
<td>Test case</td>
<td>hg38.chr4.fa</td>
<td><a href="http://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr4.fa.gz">http://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr4.fa.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-186 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
</tbody>
</table>

4.37.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Blastz software porting.
### Table 4-187 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/BLASTZ</td>
<td>Installation path of Blastz.</td>
<td>The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/CASE</td>
<td>Path for storing the test case.</td>
<td></td>
</tr>
</tbody>
</table>

### 4.37.4 Configuring the Compilation Environment

#### Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

#### Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
</tbody>
</table>

### 4.37.5 Obtaining Source Code

#### Procedure

**Step 1** Download the Blastz installation package `blastz-2004-12-27.deprecated.tar.gz` from the following link:

[http://www.bx.psu.edu/miller_lab/dist/blastz-2004-12-27.deprecated.tar.gz](http://www.bx.psu.edu/miller_lab/dist/blastz-2004-12-27.deprecated.tar.gz)

**Step 2** Run the following commands to download the `hg38.chr4.fa` and `galGal3.chr4.fa` test cases:

```
wget http://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr4.fa.gz -O hg38.chr4.fa.gz
```

**Step 3**  Use an SFTP tool to upload the downloaded source code package to the /path/to/BLASTZ directory on the server.

**Step 4**  Use an SFTP tool to upload the test cases to the /path/to/CASE directory on the server.

-----End

### 4.37.6 Compilation and Installation

**Procedure**

**Step 1**  Use PuTTY to log in to the server as the root user.

**Step 2**  Run the following command to go to the Blastz directory:

```bash
cd /path/to/BLASTZ
```

**Step 3**  Run the following command to decompress the Blastz package:

```bash
tar -xvf blastz-2004-12-27.deprecated.tar.gz
```

**Step 4**  Run the following command to switch to the directory generated after decompression:

```bash
cd blastz-source
```

**Step 5**  Run the following command to modify the compilation parameters:

```bash
sed -i 's/CFLAGS= -O/CFLAGS= -O3 -march=armv8.2-a -mtune=tsv110 -flto/g' Makefile
```

**Step 6**  Run the following command to perform compilation and installation:

```bash
CC=`which gcc` make
```

**Step 7**  Run the following command to add the Blastz environment variables:

```bash
export PATH=/path/to/BLASTZ/blastz-source:$PATH
```

**Step 8**  Run the following command to test the Blastz environment:

```bash
blastz --help
```

If information similar to the following is displayed, the environment is normal:
4.37.7 Running and Verifying Blastz

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to go to the CASE directory:

```
    cd /path/to/CASE
```

Step 3  Run the following commands to decompress the test case packages:

```
    gzip -d hg38.chr4.fa.gz
    gzip -d galGal3.chr4.fa.gz
```

Step 4  Run the following command to run the test cases:

```
    time blastz hg38.chr4.fa galGal3.chr4.fa C=3 T=2 Z=10 > hg38_c4-vs-galGal3_c4-kp7265.out
```

View the `hg38_c4-vs-galGal3_c4-kp7265.out` information.

Figure 4-21 shows an example of the result.
4.38 CAFE 5.0b2 Porting Guide (CentOS 7.6)

4.38.1 Introduction

The purpose of CAFE is to analyze changes in gene family size in a way that accounts for phylogenetic history and provides a statistical foundation for evolutionary inferences. The program uses a birth and death process to model gene gain and loss across a user-specified phylogenetic tree. The distribution of family sizes generated under this model can provide a basis for assessing the significance of the observed family size differences among taxa.

For more information about CAFE, visit the CAFE official website.

Programming language: C++

Brief description: CAFE analyzes changes in gene family size and provides a statistical foundation for evolutionary inferences.

Open-source license: ELC-2.0

Recommended Version

CAFE 5.0b2.

4.38.2 Environment Requirements

Hardware Requirements

Table 4-189 lists the hardware requirements.
Table 4-189 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-190 lists the software requirements.

Table 4-190 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAFE</td>
<td>5.0b2</td>
<td><a href="https://github.com/hahnlab/CAFExp/">https://github.com/hahnlab/CAFExp/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>archive/v5.0b2.tar.gz</td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-191 lists the OS requirements.

Table 4-191 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
</tbody>
</table>

4.38.3 Paths for Software Porting

This chapter lists the software installation paths involved in the CAFE software porting.

Table 4-192 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>
### 4.38.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages have been uploaded to the server using an SFTP tool.

#### Configuration Process

**Table 4-193 Configuration process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a></td>
</tr>
</tbody>
</table>

**4.38.5 Obtaining Source Code**

#### Procedure

**Step 1** Download the CAFE source code package `CAFExp-5.0b2.tar.gz` from the following link:

[https://github.com/hahnlab/CAFExp/archive/v5.0b2.tar.gz](https://github.com/hahnlab/CAFExp/archive/v5.0b2.tar.gz)

**Step 2** Use an SFTP tool to upload the downloaded software package to the `/path/to/CAFE` directory on the server.

----End

**4.38.6 Compilation and Installation**

#### Procedure

**Step 1** Use PuTTY to log in to the server as the `root` user.
Step 2 Run the following command to decompress the CAFE installation package:

```
   tar -xvf CAFExp-5.0b2.tar.gz
```

Step 3 Run the following command to switch to the directory generated after decompression:

```
   cd CAFExp-5.0b2
```

Step 4 Run the following command to generate the configuration file:

```
   autoconf
```

Step 5 Run the following command to perform compilation and installation:

```
   ./configure
```

Step 6 Run the following command to modify the Makefile file:

1. Run the following command to modify the Makefile file:
   
   ```
   vi Makefile
   ```

2. Press `i` to go to the edit mode and add the following content to line 7 of the file:

   ```
   -march=armv8.2-a -mtune=tsv110 -flto
   ```

3. Press `Esc`, type `:wq!`, and press `Enter` to save the file and exit.

Step 7 Run the following command to perform compilation:

```
   make -j 16
```

Step 8 Run the following command to set environment variables:

```
   export PATH=/path/to/CAFE/CAFExp-5.0b2/bin:$PATH
```

----End

### 4.38.7 Running and Verifying CAFE

#### Procedure

Step 1 Use PuTTY to log in to the server as the `root` user.

Step 2 Run the following command to switch to the directory where the test case is stored:

```
   cd /path/to/CAFE/CAFExp-5.0b2/examples
```

Step 3 Run the following commands to run the case:

```
   ../bin/cafexp -t mammals_tree.txt -i mammal_gene_families.txt -o singelambda
```

Step 4 After the commands are executed, the following information is displayed:

```
Completed 24 iterations
Time: 0H 0M 7S
Best match is: 0.0018174666100378
Final -lnL: 164769.22040255
47 values were attempted (0% rejected)
Inferring processes for Base model
Score (-lnL): 164769.22040255
```
4.38.8 More Resources

To obtain more resources, visit CAFE at GitHub: https://hahnlab.github.io/CAFE/

4.39 fastp Porting Guide (CentOS 7.6)

4.39.1 Introduction

fastp is a tool designed to provide fast all-in-one preprocessing for FastQ files. It is developed in C++ and supports multiple threads to provide high performance.

For more information about fastp, visit the fastp page at GitHub.

Programming languages: C/C++

Brief description: a fast all-in-one preprocessing for FastQ preprocessor

Open-source license: MIT license

Recommended Version

fastp 0.20.1.

4.39.2 Environment Requirements

Hardware Requirements

Table 4-194 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-195 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>fastp</td>
<td>0.20.1</td>
<td><a href="https://github.com/OpenGene/fastp/archive/v0.20.1.tar.gz">https://github.com/OpenGene/fastp/archive/v0.20.1.tar.gz</a></td>
</tr>
</tbody>
</table>
**OS Requirements**

*Table 4-196* lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

**4.39.3 Paths for Software Porting**

This chapter lists the software installation paths involved in the fastp software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see “Planning the Installation Paths” in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/FASTP</td>
<td>Installation path of fastp.</td>
<td>This installation path is only an example. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>
4.39.4 Configuring the Compilation Environment

Prerequisites
The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
</tbody>
</table>

4.39.5 Obtaining Source Code

Procedure

**Step 1** Download the fastp source code package `fastp-0.20.1.tar.gz` from the following link:

https://github.com/OpenGene/fastp/archive/v0.20.1.tar.gz

**Step 2** Use an SFTP tool to upload the source code package to the `/path/to/FASTP` directory on the server.

----End

4.39.6 Compilation and Installation

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the fastp installation package:

```
tar -xvf fastp-0.20.1.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:

```
cd fastp-0.20.1
```

**Step 4** Run the following command to perform compilation and installation:

```
make -j 16
```

**Step 5** Run the following command to set environment variables:
export PATH=/path/to/FASTP/fastp-0.20.1:$PATH

----End

4.39.7 Running and Verifying fastp

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to switch to the directory where the test case is stored:

\[ \text{cd /path/to/FASTP/testdata} \]

Step 3  Run the following commands to decompress the case files:

\[ \text{gzip -d ERR1044518_1.fastq.gz} \]
\[ \text{gzip -d ERR1044518_2.fastq.gz} \]

Step 4  Run the following commands to run the test case:

\[ \{ \text{time fastp -i ERR1044518_1.fastq -I ERR1044518_2.fastq -w 16 ;} \} 2>&1 |tee fastp.log \]

After the command is run, the following information is displayed:

- Duplication rate: 0.581538%
- Insert size peak (evaluated by paired-end reads): 169
- JSON report: fastp.json
- HTML report: fastp.html

NOTE

fastp supports multi-thread running. -w specifies the number of threads, a maximum of 16 allowed.

After the command is executed, the fast.json and fast.html reports are generated. You can use the -j and -h parameters to specify the paths and names of the two files, respectively.

To view more parameter information, run the fastp --help command.

----End

4.39.8 More Resources

To obtain more resources, visit fastp at GitHub: https://github.com/OpenGene/fastp.

4.40 GCE 1.0.2 Porting Guide (CentOS 7.6)

4.40.1 Introduction

Genomic charactor estimator (GCE) is a bayes model based method to estimate the genome size, genomic repeat content, and the heterozygisis rate of the sequencing sample. The estimated result can be used to design the sequencing strategy.
For more information about GCE, visit the [GCE page at GitHub](https://github.com/fanagislab/GCE).

Programming languages: Roff+Perl

Brief description: a bayes model based method to estimate genome characteristics

**Recommended Version**

gce-1.0.2.

**4.40.2 Environment Requirements**

**Hardware Requirements**

Table 4-199 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

**Software Requirements**

Table 2-2 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmake</td>
<td>3.17.5</td>
<td><a href="https://cmake.org/download/">https://cmake.org/download/</a></td>
</tr>
<tr>
<td>GCE</td>
<td>1.0.2</td>
<td><a href="https://github.com/fanagislab/GCE/archive/master.tar.gz">https://github.com/fanagislab/GCE/archive/master.tar.gz</a></td>
</tr>
</tbody>
</table>

**OS Requirements**

Table 4-201 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
4.40.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GCE software porting.

Table 4-202 Paths for software porting

<table>
<thead>
<tr>
<th>N o.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake.</td>
<td>The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/GCE</td>
<td>Installation path of GCE.</td>
<td></td>
</tr>
</tbody>
</table>

4.40.4 Configuring the Compilation Environment

Prerequisites

Servers must be connected to the external network. Otherwise, you need to download the software packages to the local PC and use the SFTP tool to upload the installation packages to the corresponding directories on the servers.

Configuration Process

Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing CMake</td>
<td>For details, see 4.40.4.1 Installing CMake.</td>
</tr>
</tbody>
</table>
4.40.1 Installing CMake

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to install dependencies:

```
yum install boost boost-devel openssl-devel
```

Step 3  Run the following command to decompress the CMake installation package:

```
tar -zxf cmake-3.17.5.tar.gz
```

Step 4  Run the following command to switch to the directory generated after decompression:

```
   cd cmake-3.17.5
```

Step 5  Run the following commands to perform configuration:

```
    ./configure --prefix=/path/to/CMAKE
```

Step 6  Run the following command to perform compilation and installation:

```
    make -j&& make install
```

Step 7  Run the following command to set the environment variables of CMake:

```
    export PATH=/path/to/CMAKE/bin:$PATH
```

----End

4.40.5 Obtaining Source Code

Procedure

Step 1  Download the GCE source code package from the following link:

```
https://github.com/fanagislab/GCE/archive/master.tar.gz
```

Step 2  Use an SFTP tool to upload the downloaded source code package to the /path/to/GCE directory on the server.

----End

4.40.6 Compilation and Installation

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to decompress the installation package and go to the installation directory:

```
    cd /path/to/GCE

    tar -zxf master.tar.gz&& cd GCE-master/gce-1.0.2/
```

----End
Step 3  Run the following command to delete the compiled script:

   rm -rf gce

Step 4  Run the following command to modify the configuration:

   sed -i 's/g++/g++ -O3 -march=armv8.2-a -mtune=tsv110/g' Makefile

Step 5  Run the following command to perform compilation and installation:

   make

----End

4.40.7 Running and Verifying GCE

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to switch to the installation directory of the executable file:

   cd /home/GCE/gce-1.0.2

Step 3  Run the following command to run the test:

   time ./gce -g 173854609857 -f test/Achatina_fulica/AF.kmer.freq.stat.2colum

   Figure 4-22 shows the running result.

   Figure 4-22 Running result

   ![Running result]

----End

4.40.8 More Resources

To obtain more resources, visit GCE at GitHub: https://github.com/fanagislab/GCE.

4.41 GEMMA 0.98.1 Porting Guide (CentOS 7.6)
4.41.1 Introduction

GEMMA is a software toolkit for fast application of linear mixed models (LMM) and related models to genome-wide association studies (GWAS) and other large-scale data sets.

For more information about GEMMA, visit the GEMMA page at GitHub.

Programming language: C++

Brief description: a software toolkit for fast application of linear mixed models (LMMs) and related models

Open-source license: GPL3.0

Recommended Version

GEMMA 0.98.1

4.41.2 Environment Requirements

Hardware Requirements

Table 4-203 lists the hardware requirements.

Table 4-203 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 2-2 lists the software requirements.

Table 4-204 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEMMA</td>
<td>0.98.1</td>
<td><a href="https://github.com/genetics-statistics/GEMMA/archive/0.98.1.tar.gz">https://github.com/genetics-statistics/GEMMA/archive/0.98.1.tar.gz</a></td>
</tr>
<tr>
<td>OpenBLAS</td>
<td>0.3.9</td>
<td><a href="https://github.com/xianyi/OpenBLAS/archive/v0.3.9.tar.gz">https://github.com/xianyi/OpenBLAS/archive/v0.3.9.tar.gz</a></td>
</tr>
<tr>
<td>eigen</td>
<td>3.3.8</td>
<td><a href="https://gitlab.com/libeigen/eigen/-/archive/3.3.8/eigen-3.3.8.tar.gz">https://gitlab.com/libeigen/eigen/-/archive/3.3.8/eigen-3.3.8.tar.gz</a></td>
</tr>
<tr>
<td>gsl</td>
<td>2.5.0</td>
<td><a href="https://codeload.github.com/ampl/gsl/tar.gz/v2.5.0">https://codeload.github.com/ampl/gsl/tar.gz/v2.5.0</a></td>
</tr>
</tbody>
</table>
OS Requirements

Table 4-205 lists the OS requirements.

Table 4-205 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.41.3 Paths for Software Porting

This chapter lists the software installation paths involved in the GEMMA software porting.

Table 4-206 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Installation path of each installation package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/GEMMA</td>
<td>Installation path of GEMMA.</td>
<td>The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/OpenBLAS</td>
<td>Installation path of OpenBLAS.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/EIGEN</td>
<td>Installation path of Eigen.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/GSL</td>
<td>Installation path of GSL.</td>
<td></td>
</tr>
</tbody>
</table>

4.41.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.
**Configuration Process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>Installing CMake</td>
<td>For details, see <a href="#">4.41.4.1 Installing CMake</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Installing OpenBLAS</td>
<td>For details, see <a href="#">4.41.4.2 Installing OpenBLAS</a>.</td>
</tr>
<tr>
<td>4</td>
<td>Installing Eigen</td>
<td>For details, see <a href="#">4.41.4.3 Installing Eigen</a>.</td>
</tr>
<tr>
<td>5</td>
<td>Installing GSL</td>
<td>For details, see <a href="#">4.41.4.4 Installing GSL</a>.</td>
</tr>
</tbody>
</table>

### 4.41.4.1 Installing CMake

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the CMake installation package:

```
tar zxvf cmake-3.17.3.tar.gz
```

**Step 3** Run the following command to go to the directory generated after decompression:

```
cd cmake-3.17.3
```

**Step 4** Run the following command to perform configuration:

```
./configure --prefix=/path/to/CMAKE
```

**Step 5** Run the following commands to perform compilation and installation:

```
make
make install
```

**Step 6** Run the following command to set CMake environment variables:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End
4.41.4.2 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the installation package:

```
tar xzvf OpenBLAS-0.3.9.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd OpenBLAS-0.3.9/
```

Step 4 Run the following commands to set environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 5 Run the following commands to perform compilation and installation:

```
make -j 32
make PREFIX=/path/to/OPENBLAS install
```

Step 6 Run the following commands to set OpenBLAS environment variables:

```
export LIBRARY_PATH=/path/to/OPENBLAS
export LD_LIBRARY_PATH=/path/to/OPENBLAS
```

----End

4.41.4.3 Installing Eigen

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the installation package:

```
tar xvf eigen-3.3.8.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd eigen-3.3.8
```

Step 4 Run the following commands to perform compilation and installation:

```
mkdir build
 cd build
 cmake -DCMAKE_INSTALL_PREFIX=/path/to/EIGEN ../
 make install
```

----End
4.41.4.4 Installing GSL

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the installation package:

```
tar xvf gsl-2.5.0.tar.gz
```

Step 3 Run the following command to go to the directory generated after decompression:

```
cd gsl-2.5.0
```

Step 4 Run the following commands to perform compilation and installation:

```
mkdir build

cd build

../configure --prefix=/path/to/GSL

make -j 32

make install
```

Step 5 Run the following commands to set GSL environment variables:

```
export LIBRARY_PATH=/path/to/GSL/lib:SLIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/GSL/lib:SLD_LIBRARY_PATH
export CPLUS_INCLUDE_PATH=/path/to/GSL/include:$CPLUS_INCLUDE_PATH
export C_INCLUDE_PATH=/path/to/GSL/include:$C_INCLUDE_PATH
```

----End

4.41.5 Obtaining Source Code

Procedure

Step 1 Download the GEMMA source code package GEMMA-0.98.1.tar.gz from the following link:

```
https://github.com/genetics-statistics/GEMMA/archive/0.98.1.tar.gz
```

Step 2 Use an SFTP tool to upload the source code package to the /path/to/GEMMA/ directory on the server.

----End

4.41.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install dependencies:
yum install blas-devel.aarch64 lapack-devel.aarch64 -y

Step 3  Run the following command to decompress the installation package:

tar -xvf GEMMA-0.98.1.tar.gz

Step 4  Run the following command to go to the directory generated after decompression:

cd GEMMA-0.98.1

Step 5  Run the following command to perform compilation and installation:

make EIGEN_INCLUDE_PATH=/path/to/EIGEN/include/eigen3
WITH_OPENBLAS=1 GCC_FLAGS="-Wall" LIBS="/path/to/GSL/lib/libgsl.a /path/to/GSL /lib/libgslcblas.a -L/path/to/OPENBLAS /lib -pthread -lopenblas -llapack -lblas -lz" OPENBLAS_INCLUDE_PATH=/path/to/OPENBLAS /include -j 16

Step 6  Run the following command to set environment variables:

export PATH=/path/to/GEMMA/GEMMA-0.98.1/bin:$PATH

----End

4.41.7 Running and Verifying GEMMA

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to go to the installation directory of the executable file:

cd /path/to/GEMMA/GEMMA-0.98.1/bin

Step 3  Run the following command to run the test case:

./gemma -g ../example/mouse_hs1940.geno.txt.gz -p ../example/mouse_hs1940.pheno.txt -gk -o mouse_hs1940

After the command is run, the following information is displayed:
GEMMA 0.98.1 (2018-12-10) by Xiang Zhou and team (C) 2012-2018
Reading Files ...  
## number of total individuals = 1940
## number of analyzed individuals = 1410
## number of covariates = 1
## number of phenotypes = 1
## number of total SNPs/var = 12226
## number of analyzed SNPs = 10768
Calculating Relatedness Matrix ...
================================================================== 100%
**** INFO: Done.

----End

4.41.8 More Resources

To obtain more resources, visit GEMMA at GitHub: https://github.com/genetics-statistics/GEMMA.
4.42 kmergenie 1.7051 Porting Guide (CentOS 7.6)

4.42.1 Introduction

KmerGenie analyzes the k-mer and evaluates the genome. KmerGenie conducts the automatic analysis under multiple preset k-mers. In addition to regular k-mer frequency statistics, the size of the genome can be automatically calculated based on different k-mers, and an optimal k-mer for genome assembly can be evaluated as an alternative.

Brief description: a tool for k-mer analysis and genome size evaluation.

Programing language: C++/Python

Open-source license: GPL

Recommended Version

kmergenie-1.7051

4.42.2 Environment Requirements

Hardware Requirements

Table 4-208 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-209 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>kmergenie</td>
<td>1.7051</td>
<td><a href="http://kmergenie.bx.psu.edu/kmergenie-1.7051.tar.gz">http://kmergenie.bx.psu.edu/kmergenie-1.7051.tar.gz</a></td>
</tr>
<tr>
<td>python</td>
<td>2.7.5</td>
<td>Contained in the OS</td>
</tr>
<tr>
<td>R</td>
<td>3.3.1</td>
<td><a href="https://mirrors.tuna.tsinghua.edu.cn/CRAN/src/base/R-3/R-3.3.1.tar.gz">https://mirrors.tuna.tsinghua.edu.cn/CRAN/src/base/R-3/R-3.3.1.tar.gz</a></td>
</tr>
</tbody>
</table>
### Item | Version | Download URL
--- | --- | ---
Distribute | 0.7.3 | [https://pypi.python.org/packages/source/d/distribute/distribute-0.7.3.zip --no-check-certificate](https://pypi.python.org/packages/source/d/distribute/distribute-0.7.3.zip --no-check-certificate)
Test case | human_g1k_v37 | [http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz](http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz)

### OS Requirements

*Table 4-210* lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
</table>
| CentOS | 7.6 | [https://www.centos.org/download/](https://www.centos.org/download/)
| Kernel | 4.14.0-115 | Included in the OS image. |

### 4.42.3 Planning the Paths for Software Porting

This section describes the software installation paths involved in the KmerGenie software porting.

*Table 4-211* Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/KMERGENIE</td>
<td>Installation path of KmerGenie.</td>
<td>The installation paths listed in this table are only for reference. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>4</td>
<td>/path/to/R</td>
<td>Installation path of R.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/TESTCASE</td>
<td>Path for storing the test case.</td>
<td></td>
</tr>
</tbody>
</table>
4.42.4 Configuring the Compilation Environment

Prerequisites

Installation packages are uploaded to a server using an SFTP tool.

Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

4.42.5 Obtaining the Source Code

Procedure

Step 1 Download the kmergenie-1.7051.tar.gz installation package at http://kmergenie.bx.psu.edu/kmergenie-1.7051.tar.gz

Step 2 Download test case file human_g1k_v37.fasta at http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz

Step 3 Use the SFTP tool to upload the kmergenie installation package to the /path/to/KMERGENIE directory.

Step 4 Use the SFTP tool to upload the human_g1k_v37.fasta test case to the /path/to/TESTCASE directory on the server.

----End

4.42.6 Compiling and Installing KmerGenie

4.42.6.1 Installing R

Procedure

Step 1 Run the following command to install dependencies:

```bash
yum install readline-devel libXt-devel libcurl-devel pcre-devel bzip2-devel -y
```

Step 2 Run the following command to decompress the R installation package:

```bash
tar zxvf R-3.3.1.tar.gz
```
Step 3  Run the following command to switch to the directory generated after decompression:

    cd R-3.3.1

Step 4  Run the following command to perform configuration:

    ./configure --prefix=/path/to/R

Step 5  Run the following commands to perform compilation:

    make
    make install

Step 6  Run the following command to set environment variables:

    export PATH=/path/to/R/bin:$PATH

Step 7  Run the following command to check whether R is successfully installed:

    R

    If information similar to the following is displayed, the installation is successful:

    ----End

4.42.6.2 Installing Distribute

Procedure

Step 1  Run the following command to decompress the installation package:

    unzip distribute-0.7.3.zip

Step 2  Run the following command to switch to the directory generated after decompression:

    cd distribute-0.7.3

Step 3  Run the following command to install Distribute:

    python setup.py install

    If the following information is displayed, the installation is successful:
### 4.42.6.3 Installing KmerGenie

**Procedure**

**Step 1** Run the following command to decompress the package:

```
tar zxvf kmergenie-1.7051.tar.gz
```

**Step 2** Run the following command to switch to the directory generated after decompression:

```
cd kmergenie-1.7051
```

**Step 3** Run the following command to add compilation parameters:

1. `vi makefile`
2. Press `i` to enter the editing mode and modify the eighth line in the `makefile` file as follows:

   Before the modification: `CXXFLAGS ?=-O4`
   After the modification: `CXXFLAGS ?=-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110`
3. Press `Esc`, enter `:wq!`, and press `Enter` to save the file and exit.

**Step 4** Run the following commands to perform compilation:

```
make
```

**Step 5** Run the following command to set the environment variable:

```
export PATH=/path/to/KMERGENIE/kmergenie-1.7051:$PATH
```

----End

### 4.42.7 Running and Verifying KmerGenie

**Procedure**

**Step 1** Run the following command to check the KmerGenie version:

```
kmergenie
```

The following version information is displayed:
Step 2 Run the following command to switch to the directory where the test case is stored:

cd /path/to/TESTCASE

Step 3 Run the following command to decompress the test case file:

gzip -d human_g1k_v37.fasta.gz

Step 4 Run the following command to run the test case:

kmergenie -k 200 -t 96 human_g1k_v37.fasta

If the following information is displayed and multiple .pdf files are generated, the verification is successful:

----End

4.43 LASTZ 1.04.03 Porting Guide (CentOS 7.6)

4.43.1 Introduction

LASTZ is a program for aligning DNA sequences. Originally designed to handle sequences the size of human chromosomes and from different species, it is also useful for sequences produced by Next-Generation Sequencing (NGS) technologies such as Roche 454.

For more information about LASTZ, visit the official LASTZ website.

Programing language: C/Python

One-sentence description: a program for aligning DNA sequences
Recommended Version

The recommended version is LASTZ 1.04.03.

4.43.2 Environment Requirements

Hardware Requirements

Table 4-213 lists the hardware requirements.

Table 4-213 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-214 lists the software requirements.

Table 4-214 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>LASTZ</td>
<td>1.04.03</td>
<td><a href="https://github.com/lastz/lastz/archive/1.04.03.tar.gz">https://github.com/lastz/lastz/archive/1.04.03.tar.gz</a></td>
</tr>
<tr>
<td>Test case</td>
<td>hg38.chr4.fa</td>
<td><a href="http://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr4.fa.gz">http://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr4.fa.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-215 lists the OS requirements.

Table 4-215 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
4.43.3 Planning Paths for Software Porting

This section describes the software installation paths involved in the LASTZ software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning Data for Installation&quot; in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/LASTZ</td>
<td>Installation path of LASTZ.</td>
<td>The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/CASE</td>
<td>Path for storing test cases.</td>
<td></td>
</tr>
</tbody>
</table>

4.43.4 Configuring the Compilation Environment

Prerequisites

Installation packages are uploaded to a server using an SFTP tool.

Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Basic environment setup</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
</tbody>
</table>

4.43.5 Obtaining the Source Code

Procedure

**Step 1** Download LASTZ installation package `lastz-1.04.03.tar.gz` and upload it to the `/path/to/LASTZ` directory on the server.
Download address: https://codeload.github.com/lastz/lastz/tar.gz/1.04.03

Step 2 Use the SFTP tool to upload the LASTZ source code package to the /path/to/LASTZ directory on the server.

Step 3 Run the following commands to download the hg38.chr4.fa and galGal3.chr4.fa test cases:

```
wget http://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr4.fa.gz -O hg38.chr4.fa.gz
```

Step 4 Use the SFTP tool to upload the LASTZ source code package to the /path/to/CASE directory on the server.

4.43.6 Compiling and Installing LASTZ

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the LASTZ directory:

```
cd /path/to/LASTZ
```

Step 3 Run the following command to decompress the LASTZ package:

```
tar -xvf lastz-1.04.03.tar.gz
```

Step 4 Run the following command to switch to the directory generated after decompression:

```
cd lastz-1.04.03
```

Step 5 Modify the configuration file.

1. Run the following command to modify the Makefile file:

```
vim src/Makefile
```

2. Press i to enter the editing mode and change the values of CC, definedForAll, and CFLAGS.

```
CC=/path/to/GNU/bin/gcc
definedForAll = -Wall -Wextra -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE
CFLAGS = -O3 -march=armv8.2-a -mtune=tsv110 -ftlf $(definedForAll) $(VERSION_FLAGS)
```

3. Press Esc, enter :wq!, and press Enter to save the file and exit.

Step 6 Run the following command to declare the home environment variable:

```
export HOME=/path/to/LASTZ
```

Step 7 Run the following commands to perform compilation and installation:

```
make -j16
make install
```

Step 8 Run the following command to add the LASTZ environment variable:
Step 9 Run the following command to test the LASTZ environment:

`lastz --help`

The following figure shows that the environment is normal.

---End

4.43.7 Running and Verifying LASTZ

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the CASE directory:

`cd /path/to/CASE`

Step 3 Run the following commands to decompress the test case packages:

`gzip -d hg38.chr4.fa.gz`
`gzip -d galGal3.chr4.fa.gz`

Step 4 Run the following command to run the test cases:

`time lastz hg38.chr4.fa galGal3.chr4.fa --notransition --step=20 --nogapped --format=maf > hg38_4_vs_galGal3_4-kp7265.maf`

Step 5 Run the following command to view `hg38_4_vs_galGal3_4-kp7265.maf`:

`vim hg38_4_vs_galGal3_4-kp7265.maf`

Figure 4-23 is an example of the output result.
4.43.8 Troubleshooting

Problem 1: An Error Is Reported During LASTZ Compilation

Symptom
An error is reported during LASTZ compilation. The error information is similar to "lastz.c:450:9: error: variable 'freeTargetRev' set but not used [-Werror=unused-but-set-variable]".

Possible Cause
During GCC compilation, if the -Werror parameter is configured in the Makefile file, all warnings are processed as errors.

Procedure
Delete the -Werror parameter from the Makefile file. For details, see Step 5.

4.43.9 More Resources

To obtain more resources, visit the official LASTZ website:

https://github.com/lastz/lastz/tree/1.04.03

4.44 MAFFFT Porting Guide (CentOS 7.6)

4.44.1 Introduction

MAFFT is a multiple sequence alignment program for UNIX-like operating systems. It offers a range of multiple alignment methods, L-INS-i (accurate; for alignment of <~200 sequences), FFT-NS-2 (fast; for alignment of <~30,000 sequences), etc.
For more information about MAFFT, visit the [official MAFFT website](https://mafft.cbrc.jp/alignment/software).

Programming language: C

One-sentence description: a program used to create multiple sequence alignments of amino acid or nucleotide sequences based on the Fast Fourier Transform

Open-source license: Apache License 2.0

**Recommended Version**

The recommended version is 7.470 or later.

### 4.44.2 Environment Requirements

**Hardware Requirements**

Table 4-218 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

**Software Requirements**

Table 4-219 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test case</td>
<td>LSU rRNA</td>
<td><a href="https://mafft.cbrc.jp/alignment/software/ex1.txt">https://mafft.cbrc.jp/alignment/software/ex1.txt</a></td>
</tr>
</tbody>
</table>

**OS Requirements**

Table 4-220 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
</tbody>
</table>
### 4.44.3 Planning Paths for Software Porting

This section describes the software installation paths involved in the MAFFT software porting.

#### Table 4-221 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/MAFFT</td>
<td>Installation path of MAFFT.</td>
<td>The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/CASE</td>
<td>Path for storing test cases.</td>
<td></td>
</tr>
</tbody>
</table>

### 4.44.4 Obtaining the Source Code

**Procedure**

**Step 1** Download MAFFT installation packages `mafft-7.471-with-extensions-src.tgz` and `mafft-7.471-without-extensions-src.tgz` and upload them to the `/path/to/MAFFT` directory on a server.

Download addresses:


**Step 2** Use the SFTP tool to upload the MAFFT installation package to the `/path/to/MAFFT` directory on the server.

----End
4.44.5 Compiling and Installing MAFFT

4.44.5.1 Basic Installation

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to create a main program installation directory:

```bash
mkdir -p /path/to/MAFFT
```

**Step 3** Run the following command to copy the installation package to the main program installation directory:

```bash
cp mafft-7.471-without-extensions-src.tgz /path/to/MAFFT
```

**Step 4** Run the following command to go to the main program installation directory:

```bash
cd /path/to/MAFFT
```

**Step 5** Run the following command to decompress the installation package:

```bash
tar -xvf mafft-7.471-without-extensions-src.tgz
```

**Step 6** Run the following command to switch to the directory generated after decompression:

```bash
cd mafft-7.471-without-extensions/core
```

**Step 7** Run the following commands to configure related files:

```bash
perl -p -i -e 's#PREFIX =.*#PREFIX = /PATH/to/MAFFT#'
Makefile
```

```bash
sed -i 's/gcc/`which gcc`/g'
Makefile
```

```bash
sed -i 's/-O3/-O3 -march=armv8.2-a -mtune=tsv110 -flto/g'
Makefile
```

**Step 8** Compile and install the main program of MAFFT.

1. Run the following commands to compile and install the main program:

   ```bash
   make clean
   make
   make install
   ```

2. Run the following command to add the environment variable:

   ```bash
   export PATH=/path/to/MAFFT/bin:$PATH
   ```

----End

4.44.5.2 (Optional) Installing MPI

Procedure

**Step 1** Run the following command to switch to the directory generated after decompression:
Step 2 Run the following commands to configure related files:
perl -p -i -e 's#PREFIX =.*#PREFIX = /path/to/MAFFT #'
Makefile
sed -i 's/-O3/-O3 -march=armv8.2-a -mtune=tsv110 -flto/g'
Makefile

Step 3 Run the following commands to compile and install the MAFFT extensions:
make clean
make
make install

4.44.5.3 (Optional) Installing MAFFT Extensions

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to copy the installation package to the main program installation directory:
cp mafft-7.471-with-extensions-src.tgz /path/to/MAFFT

Step 3 Run the following command to decompress the installation package:
tar -xvf mafft-7.471-with-extensions-src.tgz

Step 4 Run the following command to switch to the directory generated after decompression:

cd /path/to/MAFFT/mafft-7.471-with-extensions/extensions

Step 5 Run the following command to configure related files:
perl -p -i -e 's#PREFIX =.*#PREFIX = /PATH/to/MAFFT #'
Makefile

Step 6 Run the following commands to compile and install the MAFFT extensions:
make clean
make
make install

4.44.6 Running and Verifying MAFFT

Procedure

Step 1 Run the following command to switch to the test directory.

cd /path/to/CASE
Step 2 Run the following command to add the environment variable:

```bash
export PATH=/path/to/MAFFT/bin:$PATH
```

Step 3 Run the following command to check whether the software is correctly installed:

```bash
{ time MAFFT_N_THREADS_PER_PROCESS="1" MAFFT_MPIRUN="mpirun --allow-run-as-root -n 128" MAFFT_TMPDIR="/path/to/CASE" mafft --mpi --large --globalpair ex1.txt > kp_out; } 2>&1 | tee -a `hostname`_`date '+%Y%m%d%H%M%S'`.log
```

The following figure shows the execution result:

```
Progressive alignment ...
STEP  54 / 58
Reallocating...done. *alloclen = 11410
STEP  58 / 58
done.
tbfast (nuc) Version 7.471
alg=A, model=DNA200 (2), 1.53 (4.59), -0.00 (-0.00), noshift, amax=0.0
1 thread(s)

Strategy:
G-large-INS-1 (Not tested.)
?

If unsure which option to use, try 'mafft --auto input > output'.
For more information, see 'mafft --help', 'mafft --man' and the mafft page.

The default gap scoring scheme has been changed in version 7.110 (2013 Oct)
It tends to insert more gaps into gap-rich regions than previous versions.
To disable this change, add the --leavegappyregion option.

real  0m38.018s
user  37m53.309s
sys   3m43.047s
```

----End

4.44.7 More Resources

To obtain more resources, visit the official website of MAFFT:

https://mafft.cbrc.jp/alignment/software/

4.45 megahit 1.2.9 Porting Guide (CentOS 7.6)

4.45.1 Introduction

MEGAHIT is an ultra-fast and memory-efficient Next-Generation Sequencing (NGS) assembler. It is optimized for metagenomes, but also works well on generic single genome assembly and single-cell assembly.

Programming language: C++/C/Python

Open-source license: GPL3.0
Recommended Version

The recommended version is megahit 1.2.9.

4.45.2 Environment Requirements

Hardware Requirements

Table 4-222 lists the hardware requirements.

Table 4-222 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-223 lists the software requirements.

Table 4-223 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEGAHIT</td>
<td>1.2.9</td>
<td><a href="https://codeload.github.com/voutcn/megahit/tar.gz/v1.2.9">https://codeload.github.com/voutcn/megahit/tar.gz/v1.2.9</a></td>
</tr>
<tr>
<td>Test case 1</td>
<td>SRR1976948</td>
<td><a href="https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/">https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/</a> SRR1976948.abundtrim.subset.pe.fq.gz</td>
</tr>
<tr>
<td>Test case 2</td>
<td>SRR1977249</td>
<td><a href="https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/">https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/</a> SRR1977249.abundtrim.subset.pe.fq.gz</td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-224 lists the OS requirements.

Table 4-224 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
4.45.3 Planning Paths for Software Porting

This section describes the software installation paths involved in the MEGAHIT software porting.

Table 4-225 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/MEGAHIT</td>
<td>Installation path of MEGAHIT.</td>
<td>The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

4.45.4 Configuring the Compilation Environment

Prerequisites

Servers must be connected to the external network. Otherwise, you need to download the software packages to the local PC and use the SFTP tool to upload the installation packages to the corresponding directories on the servers.

Configuration Process

Table 4-226 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Basic environment setup</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>CMake installation</td>
<td>For details, see 4.45.4.1 Installing CMake.</td>
</tr>
</tbody>
</table>
4.45.4.1 Installing CMake

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to install dependency packages, such as the CMake package:

```
yum install -y cmake gzip zlib bzip2
```

----End

4.45.5 Obtaining the Source Code

Procedure

Step 1  Download the MEGAHIT source code from the following address:

```
https://codeload.github.com/voutcn/megahit/tar.gz/v1.2.9
```

Step 2  Use the SFTP tool to upload the MEGAHIT source code package to the /path/to/MEGAHIT directory on the server.

----End

4.45.6 Compiling and Installing MEGAHIT

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to go to the source code installation directory:

```
cd /path/to/MEGAHIT
```

Step 3  Run the following command to decompress the source code package:

```
tar -xvf v1.2.9.tar.gz
```

Step 4  Run the following command to go to the folder generated after the decompression:

```
cd megahit-1.2.9
```

Step 5  Run the following command to modify the CMakeLists.txt file:

```
 vim CMakeLists.txt
```

Before the modification:

```
6 set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11")
...
76 set_target_properties(megahit_core PROPERTIES COMPILERSFLAGS "-mbmi2 -DUSE_BMI2 -mpopcnt")
77 set_target_properties(megahit_core_popcnt PROPERTIES COMPILERSFLAGS "-mpopcnt")
```

After the modification:

```
6 set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11 -march=armv8.2-a")
...
76 set_target_properties(megahit_core PROPERTIES COMPILERSFLAGS "")
77 set_target_properties(megahit_core_popcnt PROPERTIES COMPILERSFLAGS "")
```
Step 6  Run the following command to switch the directory:

```
cd ../src/utils/
```

Step 7  Use the SFTP tool to upload the two files to the directory.

```
KunpengTran.h  cpu_dispatch.h
```

Step 8  Obtain the `sse2neon.h` file from the following address:

```
https://github.com/DLTcollab/sse2neon/blob/master/sse2neon.h
```

Step 9  Run the following command to copy the Neon instruction conversion code to a specific directory of the compiler:

```
cp sse2neon.h /path/to/GNU/gcc/include
```

Step 10  Run the following command to switch the directory:

```
cd ../kmlib/
```

Step 11  Run the `vi` command to open the text file and modify the code:

```
vim kmrns.h
```

Before the modification:
```
#include <x86intrin.h>
```

After the modification:
```
#include <sse2neon.h>
```

Step 12  Run the following command to create a build folder and go to the folder:

```
mkdir /path/to/MEGAHIT/build && cd /path/to/MEGAHIT/build
```

Step 13  Run the following commands to perform compilation and installation:

```
cmake -DCMAKE_INSTALL_PREFIX=/path/to/MEGAHIT -
DCMAKE_BUILD_TYPE=Release ../
making -j 16
make install
```

Step 14  Run the following command to load the environment variable:

```
extport PATH=/path/to/MEGAHIT/bin:SPATH
```

----End

4.45.7 Running and Verifying MEGAHIT

Procedure

Step 1  Use PuTTY to log in to the server as the `root` user.

Step 2  Run the following command to go to the source code installation directory:

```
cd /path/to/MEGAHIT
```
Step 3 Run the following command to create a test case folder and go to the folder:

```
mkdir testcase && cd testcase
```

Step 4 Upload the downloaded test case file to the current directory.

Step 5 Run the following command to run the test case:

```
megahit -t 128 --12 SRR1976948.abundtrim.subset.pe.fq.gz
SRR1977249.abundtrim.subset.pe.fq.gz > megahit.log
```

Step 6 After the execution is complete, check the value of Time elapsed in the `megahit.log` file. The unit is seconds. A smaller value indicates better performance. The following is a sample of the output result.

<table>
<thead>
<tr>
<th>Time</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020-11-19</td>
<td>Merging to output final contigs</td>
</tr>
<tr>
<td>2020-11-19</td>
<td>3227 contigs, total 10298772 bp, min 200 bp, max 301573 bp, avg 3191 bp, N50 6898 bp</td>
</tr>
<tr>
<td>2020-11-19</td>
<td>ALL DONE. Time elapsed: 126.348002 seconds</td>
</tr>
</tbody>
</table>

---End

4.45.8 More Resources

To obtain more resources, visit the MEGAHIT page of GitHub:

https://github.com/voutcn/megahit

4.46 PASA 2.4.1 Porting Guide (CentOS 7.6)

4.46.1 Introduction

Program to Assemble Spliced Alignments (PASA) is a eukaryotic genome annotation tool that exploits spliced alignments of expressed transcript sequences to automatically model gene structures and to maintain gene structure annotation consistent with the most recently available sequence data. PASA can also identify and classify all splicing variations supported by the transcript alignments.

For more information, visit the PASA official website.

Programming language: Perl, or C++

Brief description: A eukaryotic genome annotation tool

Recommended Version

PASA 2.4.1

4.46.2 Environment Requirements

Hardware Requirements

Table 4-227 lists the hardware requirements.
Table 4-227 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-228 lists the software requirements.

Table 4-228 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASA</td>
<td>2.4.1</td>
<td><a href="https://github.com/PASApipeline/PASApipeline/">https://github.com/PASApipeline/PASApipeline/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>archive/pasa-v2.4.1.tar.gz</td>
</tr>
<tr>
<td>URI</td>
<td>1.35</td>
<td><a href="http://search.cpan.org/CPAN/authors/id/G/GA/">http://search.cpan.org/CPAN/authors/id/G/GA/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GAAS/URI-1.35.tar.gz</td>
</tr>
<tr>
<td>blat</td>
<td>35.1</td>
<td><a href="https://codeload.github.com/djhshih/blat/tar.gz/">https://codeload.github.com/djhshih/blat/tar.gz/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>v35.1</td>
</tr>
<tr>
<td>gmap</td>
<td>2015-09-21</td>
<td><a href="http://research-pub.gene.com/gmap/src/gmap-">http://research-pub.gene.com/gmap/src/gmap-</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>gsnap-2015-09-21.tar.gz</td>
</tr>
<tr>
<td>samtoo ls</td>
<td>0.1.9</td>
<td><a href="https://codeload.github.com/samtools/samtools/">https://codeload.github.com/samtools/samtools/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>tar.gz/0.1.9</td>
</tr>
<tr>
<td>mario db</td>
<td>5.5.60</td>
<td>Contained in the CentOS 7.6 image</td>
</tr>
<tr>
<td>Test case</td>
<td>genome_sample</td>
<td>Provided by the software</td>
</tr>
<tr>
<td></td>
<td>.fasta</td>
<td></td>
</tr>
<tr>
<td></td>
<td>all_transcripts.fasta</td>
<td></td>
</tr>
<tr>
<td></td>
<td>all_transcripts.fasta.clean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>all_transcripts.fasta</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FL_accs.txt</td>
<td></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-229 lists the OS requirements.

Table 4-229 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
</tbody>
</table>
### 4.46.3 Planning the Paths for Software Porting

This section describes the software installation paths involved in the PASA software porting.

**Table 4-230 Paths for software porting**

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning Data for Installation&quot; in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/BLAT</td>
<td>Installation path of Blat.</td>
<td>The installation paths listed in this table are only for reference. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/URI</td>
<td>Installation path of URI.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/PASA</td>
<td>Installation path of PASA.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/SAMTOOLS</td>
<td>Installation path of SAMtools.</td>
<td>For details about the installation and compilation, see the <a href="#">gmap 2015.9.21 Porting Guide</a>(CentOS 7.6).</td>
</tr>
<tr>
<td>6</td>
<td>/path/to/GMAP</td>
<td>Installation path of GMAP.</td>
<td></td>
</tr>
</tbody>
</table>

### 4.46.4 Configuring the Compilation Environment

#### Prerequisites

Installation packages are uploaded to a server using an SFTP tool.
Configuration Process

Table 4-231  Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install Perl.</td>
<td>For details, see 4.46.4.1 Installing Perl.</td>
</tr>
<tr>
<td>3</td>
<td>Install URI.</td>
<td>For details, see 4.46.4.2 Installing URI.</td>
</tr>
<tr>
<td>4</td>
<td>Install Blat.</td>
<td>For details, see 4.46.4.3 Installing Blat.</td>
</tr>
<tr>
<td>5</td>
<td>Install MariaDB.</td>
<td>For details, see 4.46.4.4 Installing MariaDB.</td>
</tr>
<tr>
<td>6</td>
<td>Install SAMtools.</td>
<td>For details, see 4.46.4.5 Installing SAMtools.</td>
</tr>
</tbody>
</table>

4.46.4.1 Installing Perl

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to install Perl:

`yum install perl* -y`

----End

4.46.4.2 Installing URI

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to decompress the URI package:

`tar -xvf URI-1.35.tar.gz`

Step 3  Run the following command to switch to the directory storing the decompressed file:

`cd URI-1.35`

Step 4  Run the following commands to perform compilation and installation:

`perl Makefile.PL`
make
make install
----End

4.46.4.3 Installing Blat

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to install dependencies:

```
yum install libpng-devel -y
```

Step 3  Run the following command to decompress the Blat package:

```
tar zxvf blat-35.1.tar.gz
```

Step 4  Run the following command to switch to the directory storing the decompressed file:

```
cd blat-35.1/
```

Step 5  Run the following command to declare the installation environment:

```
export MACHTYPE=aarch64
```

Step 6  Run the following command to perform compilation and installation:

```
make
```

Step 7  Run the following command to declare the Blat environment variable:

```
export PATH=/path/to/BLAT/blat-35.1/bin:SPATH
```

----End

4.46.4.4 Installing MariaDB

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to install MariaDB:

```
yum install mariadb* -y
```

Step 3  Run the following command to modify the my.cnf file:

```
vim /etc/my.cnf
```

Before the modification:

```
[mysqld]
```

After the modification:

```
[mysqld]
skip-grant-tables
```
**Step 4** Run the following command to start the database:

```bash
systemctl start mariadb.service
```

**Step 5** Run the following command to access the database:

```bash
mysql -u root
```

**Step 6** Run the following command to access MySQL:

```sql
use mysql;
```

**Step 7** Run the following commands to modify the plugin attribute:

```sql
update user set authentication_string=PASSWORD(''),
plugin='mysql_native_password' where user='root';
flush privileges;
```

**Step 8** Run the following commands to create a user with the read-only permission and a user with all permissions:

```sql
MariaDB [(none)]> grant select on *.* to 'test'@'%' identified by '123456';
MariaDB [(none)]> grant all on *.* to 'pasa'@'%' identified by '123456';
MariaDB [(none)]> flush privileges;
```

**Step 9** After the creation is complete, enter `exit`.

**Step 10** Run the following command to check whether the users are created:

```bash
mysql -upasa -p123456
```

If the following information is displayed, the creation is successful:

```
[root@ARMCI sample data]# mysql -upasa -p123456
Welcome to the MariaDB monitor. Commands end with ; or \\g.  
Your MariaDB connection id is 49 
Server version: 5.5.63-MariaDB MariaDB Server 
Copyright (C) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement. 
```

---End

# 4.46.4.5 Installing SAMtools

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to install dependencies:

```bash
yum install ncurses-devel.aarch64 bzip2-devel.aarch64 xz-devel.aarch64 java-1.8.0-openjdk-devel.aarch64 -y
```

**Step 3** Run the following command to decompress the SAMtools package:

```bash
tar -xvf samtools-0.1.9.tar.gz
```
Step 4 Run the following command to switch to the directory generated after decompression:

```bash
cd samtools-0.1.9
```

Step 5 Run the following command to obtain the absolute path of GNU:

```bash
which gcc
```

Step 6 Run the following command to modify the `Makefile` file:

```bash
vi Makefile
```

Before the modification:

| CC | gcc |

After the modification:

| CC | path/to/GNU/bin/gcc |

Step 7 Run the following command to perform compilation and installation:

```bash
make
```

Step 8 Run the following command to add the environment variable of SAMtools:

```bash
export PATH=/path/to/SAMTOOLS/samtools-0.1.9:SPATH
```

---- End

### 4.46.5 Obtaining the Source Code

**Procedure**

**Step 1** Download the PASA installation package `pasa-v2.4.1.tar.gz`.

- Download address: [https://github.com/PASApipeline/PASApipeline/archive/pasa-v2.4.1.tar.gz](https://github.com/PASApipeline/PASApipeline/archive/pasa-v2.4.1.tar.gz)

**Step 2** Use the SFTP tool to upload the PASA package to the `/path/to/PASA` directory.

---- End

### 4.46.6 Compiling and Installing PASA

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the PASA package:

```bash
tar -xvf pasa-v2.4.1.tar.gz
```

**Step 3** Run the following command to switch to the directory generated after decompression:

```bash
cd PASApipeline-pasa-v2.4.1
```

**Step 4** Run the following command to perform compilation and installation:

```bash
make -j16
```
**4.46.7 Running and Verifying PASA**

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to add the GMAP environment variable:

```bash
export PATH=/path/to/GMAP/Install/bin:$PATH
```

**Step 3** Run the following commands to modify configuration files:

```bash
cp /path/to/PASA/PASApipeline-pasa-v2.4.1/pasa_conf/pasa.CONFIG.template /path/to/PASA/PASApipeline-pasa-v2.4.1/pasa_conf/conf.txt
vim /path/to/PASA/PASApipeline-pasa-v2.4.1/pasa_conf/conf.txt
```

```
MYSQLSERVER=localhost
MYSQL_RO_USER=test
MYSQL_RO_PASSWORD=123456
MYSQL_RW_USER=pasa
MYSQL_RW_PASSWORD=123456
```

Enter `:wq!` to save the change and exit.

**Step 4** Run the following command to switch to the directory where the test case is stored:

```bash
cd /path/to/PASA/PASApipeline-pasa-v2.4.1/sample_data
```

**Step 5** Run the following commands to modify configuration files:

```bash
cp /path/to/PASA/PASApipeline-pasa-v2.4.1/pasa_conf/pasa.alignAssembly.Template.txt /path/to/PASA/PASApipeline-pasa-v2.4.1/pasa_conf/alignAssembly.config
vi alignAssembly.config
```

```
DATABASE=sample_data
```

Enter `:wq!` to save the change and exit.

**Step 6** Run the following commands to run the test case:

```bash
gzip -d genome_sample.fasta.gz
{ time Launch_PASA_pipeline.pl -c alignAssembly.config -C -R -g genome_sample.fasta -t all_transcripts.fasta.clean -T -u all_transcripts.fasta -f FL_accs.txt --ALIGNERS blat,gmap --CPU 96; } 2>&1 |tee PASA.log
```

View the `PASA.log` file. If **Finished** is recorded in the log file, the PASA process ends properly.

**Figure 4-24** shows an example of the running result.
4.46.8 Troubleshooting

Problem 1: An error is reported when MariaDB verifies the user.

Symptom
An error message similar to "ERROR 1045 (28000): Access denied for user 'test'@'localhost' (using password: YES)" is displayed when MariaDB verifies the user.

Possible Cause
The default plugin attribute is incorrect.

Procedure:
See Step 7.

Problem 2: An error is reported when MariaDB creates a database.

Symptom
MariaDB verifies the user successfully, but an error message similar to "ERROR 1045 (28000): Access denied for user 'pasa'@'localhost' (using password: YES)" during database creation.

Possible Cause

The permission of the paas user is incorrect.

Procedure:

Run the following commands:

```
mysql -upasa -p123456
use mysql;
insert into mysql.user(Host,User,Password)
values("localhost","pasa",password("123456"));
```

Problem 3: An error is reported during the PASA running.

Symptom

An error message similar to "Can't locate URI/Escape.pm in @INC" is displayed during the PASA running.

Possible Cause

The URI is missed in Perl.

Procedure

See 4.46.4.2 Installing URI.

4.46.9 More Resources

PASA official website:

https://github.com/PASApipeline/PASApipeline/wiki

4.47 Augustus 3.3.3 Porting Guide (CentOS 7.6)

4.47.1 Introduction

AUGUSTUS is an open-source program that predicts genes in eukaryotic genomic sequences. It is the most accurate \textit{ab initio} program whose accuracy is dependent on the quality of the training gene sets used for specific parameters of species.

For more information, please visit http://bioinf.uni-greifswald.de/augustus/.

Programming language: C++

Brief description: Gene prediction

Recommended Version

Augustus 3.3.3
4.47.2 Environment Requirements

Hardware Requirements

Table 4-232 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-233 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmake</td>
<td>3.8.1</td>
<td><a href="https://cmake.org/download/">https://cmake.org/download/</a></td>
</tr>
<tr>
<td>htslib</td>
<td>1.11</td>
<td><a href="https://github.com/samtools/htslib/archive/1.11.tar.gz">https://github.com/samtools/htslib/archive/1.11.tar.gz</a></td>
</tr>
<tr>
<td>bcftools</td>
<td>1.11</td>
<td><a href="https://github.com/samtools/bcftools/archive/1.11.tar.gz">https://github.com/samtools/bcftools/archive/1.11.tar.gz</a></td>
</tr>
<tr>
<td>samtools</td>
<td>1.11</td>
<td><a href="https://github.com/samtools/samtools/archive/1.11.tar.gz">https://github.com/samtools/samtools/archive/1.11.tar.gz</a></td>
</tr>
<tr>
<td>bamtools</td>
<td>2.5.0</td>
<td><a href="https://github.com/pezmaster31/bamtools/archive/v2.5.0.tar.gz">https://github.com/pezmaster31/bamtools/archive/v2.5.0.tar.gz</a></td>
</tr>
<tr>
<td>Augustus</td>
<td>3.3.3</td>
<td><a href="https://github.com/Gaius-Augustus/Augustus/archive/v3.3.3.tar.gz">https://github.com/Gaius-Augustus/Augustus/archive/v3.3.3.tar.gz</a></td>
</tr>
<tr>
<td>Test case</td>
<td>sag178</td>
<td><a href="http://bioinf.uni-greifswald.de/augustus/datasets/sag178.gb.gz">http://bioinf.uni-greifswald.de/augustus/datasets/sag178.gb.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-234 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
4.47.3 Planning the Paths for Software Porting

This section describes the software installation paths involved in the AUGUSTUS software porting.

Table 4-235 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning Data for Installation&quot; in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/AUGUSTUS/htslib</td>
<td>Installation path of HTSlib.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>/path/to/AUGUSTUS/bcftools</td>
<td>Installation path of BCFtools.</td>
<td>The installation paths listed in this table are only for reference. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>4</td>
<td>/path/to/AUGUSTUS/samtools</td>
<td>Installation path of SAMtools.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/AUGUSTUS/cmake</td>
<td>Installation path of CMake.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/AUGUSTUS/bamtools</td>
<td>Installation path of Bamtools.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>/path/to/AUGUSTUS/augustus</td>
<td>Installation path of AUGUSTUS.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>/path/to/AUGUSTUS/CASE</td>
<td>Path for storing cases.</td>
<td></td>
</tr>
</tbody>
</table>

4.47.4 Configuring the Compilation Environment

**Prerequisites**

Installation packages are uploaded to a server using an SFTP tool.
### Configuration Process

#### Table 4-236 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Single-Node Scenario&quot; in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>Install CMake.</td>
<td>For details, see <a href="#">4.47.4.1 Installing CMake</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Install HTSlib.</td>
<td>For details, see <a href="#">4.47.4.2 Installing HTSlib</a>.</td>
</tr>
<tr>
<td>4</td>
<td>Install BCFtools.</td>
<td>For details, see <a href="#">4.47.4.3 Installing BCFtools</a>.</td>
</tr>
<tr>
<td>5</td>
<td>Install SAMtools.</td>
<td>For details, see <a href="#">4.47.4.4 Installing SAMtools</a>.</td>
</tr>
<tr>
<td>6</td>
<td>Install Bamtools.</td>
<td>For details, see <a href="#">4.47.4.5 Installing Bamtools</a>.</td>
</tr>
</tbody>
</table>

#### 4.47.4.1 Installing CMake

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Use the SFTP tool to upload the CMake installation package to the /path/to/CMAKE directory.

**Step 3** Run the following command to decompress the CMake installation package:

```
  cd /path/to/AUGUSTUS && tar -zxf cmake-3.8.1.tar.gz
```

**Step 4** Run the following command to switch to the directory generated after decompression:

```
  cd cmake-3.8.1
```

**Step 5** Run the following command to perform configuration:

```
  ./configure --prefix=/path/to/AUGUSTUS/cmake
```

**Step 6** Run the following commands to perform compilation and installation:

```
  make
  make install
```

**Step 7** Run the following command to set CMake environment variables:
export PATH=/path/to/cmake/bin:$PATH

----End

4.47.4.2 Installing HTSlib

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Use the SFTP tool to upload the HTSlib installation package to the /path/to/AUGUSTUS directory.
Step 3 Run the following command to install dependencies:

`yum install -y libcurl libcurl-devel openssl openssl-devel bzip2 bzip2-devel xz-devel boost-devel`

Step 4 Run the following command to decompress the package:

`cd /path/to/AUGUSTUS && tar -zxf htslib-1.11.tar.gz`

Step 5 Run the following command to rename and go to the installation directory:

`mv htslib-1.11 htslib && cd htslib`

Step 6 Run the following commands to perform configuration:

`autoheader`
`autoconf`
`sed -i 's/-O2/-O3 -march=armv8.2-a -mtune=tsv110/g' `grep -lr "-O2" .`/` CC=`which gcc` ./configure --prefix=/path/to/AUGUSTUS/htslib`

Step 7 Run the following commands to perform compilation and installation:

`make -j 16`
`make install`

----End

4.47.4.3 Installing BCFtools

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Use the SFTP tool to upload the BCFtools installation package to the /path/to/AUGUSTUS directory.
Step 3 Run the following command to decompress the package:

`cd /path/to/AUGUSTUS && tar -zxf bcftools-1.11.tar.gz`

Step 4 Run the following command to rename and go to the installation directory:

`mv bcftools-1.11 bcftools && cd bcftools`
Step 5 Run the following commands to perform configuration:

- autoheader
- autoconf
- `sed -i 's/-O2/-O3 -march=armv8.2-a -mtune=tsv110/g' `grep -lr "-O2" ./`
- `CC=`which gcc` ./configure --prefix=/path/to/AUGUSTUS/bcftools`

Step 6 Run the following commands to perform compilation and installation:

- `make -j 16`
- `make install`

Step 7 Run the following commands to perform configuration:

- `sed -i 's/-O2/-O3 -march=armv8.2-a -mtune=tsv110/g' `grep -lr "-O2" ./`
- `CC=`which gcc` ./configure --prefix=/path/to/AUGUSTUS/samtools`

Step 8 Run the following commands to perform compilation and installation:

- `make -j 16`
- `make install`
4.47.4.5 Installing Bamtools

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Use the SFTP tool to upload the Bamtools installation package to the /path/to/AUGUSTUS directory.

Step 3  Run the following command to decompress the package:

```
  cd /path/to/AUGUSTUS && tar -zxf bamtools-2.5.0.tar.gz
```

Step 4  Run the following command to rename and go to the installation directory:

```
  mv bamtools-2.5.0 bamtools && cd bamtools
```

Step 5  Run the following commands to perform configuration:

```
  mkdir build && cd build
  cmake -DCMAKE_INSTALL_PREFIX=../..
```

Step 6  Run the following commands to perform compilation and installation:

```
  make -j 16
  make install
```

----End

4.47.5 Obtaining the Source Code

Procedure

Step 1  Download the Augustus installation package Augustus-3.3.3.tar.gz at

  https://github.com/Gaius-Augustus/Augustus/archive/v3.3.3.tar.gz

Step 2  Use the SFTP tool to upload the AUGUSTUS installation package to the /path/to/AUGUSTUS directory.

----End

4.47.6 Compiling and Installing AUGUSTUS

Procedure

Step 1  Run the following command to go to the installation directory and decompress the software package:

```
  cd /path/to/AUGUSTUS && tar -zxf Augustus-3.3.3.tar.gz && cd Augustus-3.3.3
```

Step 2  Run the following commands to perform configuration:

```
  export TOOLDIR=/path/to/AUGUSTUS
  export LD_LIBRARY_PATH=/path/to/AUGUSTUS/bamtools/lib64:SLD_LIBRARY_PATH
```

Issue 14 (2021-07-23)  Copyright © Huawei Technologies Co., Ltd.  698
export LIBRARY_PATH=/path/to/AUGUSTUS/bamtools/lib64:$LIBRARY_PATH

sed -i 's//usr/include/bamtools//path/to/AUGUSTUS/bamtools/include/bamtools/g' `grep -rl "/usr/include/bamtools" ./`

Step 3 Run the following command to set the installation path:

```
sed -i 's//opt/augustus//path/to/AUGUSTUS/augustus_aarch64/g'
```

Step 4 Run the following commands to perform compilation and installation:

```
CC=`which gcc` CXX=`which g++` CXXFLAGS='-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110' CFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' make -j
make install
```

----End

4.47.7 Running and Verifying AUGUSTUS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to set environment variables:

```
export PATH=/path/to/AUGUSTUS/augustus_aarch64-3.3.3/bin:$PATH
```

Step 3 Run the following commands to run the test case:

```
cd /path/to/AUGUSTUS/CASE

{ time augustus --species=human --UTR=on sag178.gb; } 2>&1 | tee -a `hostname`_'date '+%Y%m%d%H%M%S'`.log
```

Figure 4-25 shows an example of the running result.
4.48 Delly 0.8.5 Porting Guide (CentOS 7.6)

4.48.1 Introduction

Delly is an integrated structural variant (SV) prediction method that can discover, genotype and visualize deletions, tandem duplications, inversions and translocations at single-nucleotide resolution in short-read massively parallel sequencing data. It uses paired-ends, split-reads, and read-depth to sensitively and accurately delineate genomic rearrangements throughout the genome.

For more information about Delly, visit https://tobiasrausch.com/delly/.

Programming language: C++

Brief description: an integrated structural variant (SV) prediction method

Recommended Version

Delly 0.8.5
4.48.2 Environment Requirements

Hardware Requirements

Table 4-237 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-238 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>boost</td>
<td>1.66</td>
<td><a href="https://dl.bintray.com/boostorg/release/1.66.0/source/boost_1_66_0.tar.gz">https://dl.bintray.com/boostorg/release/1.66.0/source/boost_1_66_0.tar.gz</a></td>
</tr>
<tr>
<td>samtools</td>
<td>1.10</td>
<td><a href="https://github.com/samtools/samtools/archive/1.10.tar.gz">https://github.com/samtools/samtools/archive/1.10.tar.gz</a></td>
</tr>
<tr>
<td>bcftools</td>
<td>1.10.2</td>
<td><a href="https://github.com/samtools/bcftools/releases/download/1.10.2/bcftools-1.10.2.tar.bz2">https://github.com/samtools/bcftools/releases/download/1.10.2/bcftools-1.10.2.tar.bz2</a></td>
</tr>
<tr>
<td>bwa</td>
<td>0.7.17</td>
<td><a href="https://sourceforge.net/projects/bio-bwa/files/">https://sourceforge.net/projects/bio-bwa/files/</a></td>
</tr>
<tr>
<td>delly</td>
<td>0.8.5</td>
<td><a href="https://github.com/dellytools/delly/archive/v0.8.5.tar.gz">https://github.com/dellytools/delly/archive/v0.8.5.tar.gz</a></td>
</tr>
<tr>
<td>htslib</td>
<td>1.10.2</td>
<td><a href="https://github.com/samtools/htslib/archive/1.10.2.tar.gz">https://github.com/samtools/htslib/archive/1.10.2.tar.gz</a></td>
</tr>
<tr>
<td></td>
<td>B17NC_R1.fq.gz</td>
<td><a href="https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz">https://obs.cn-north-1.myhuaweicloud.com/obs-82bb/data/Panel/B17NC_R1.fastq.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-239 lists the OS requirements.
Table 4-239 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.48.3 Paths for Software Porting

This chapter lists the software installation paths involved in the Delly software porting.

Table 4-240 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each installation package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/DELLY/delly-0.8.5</td>
<td>Installation path of Delly.</td>
<td>The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/DELLY/boost</td>
<td>Installation path of Boost.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/DELLY/CASE</td>
<td>Path for storing test cases.</td>
<td></td>
</tr>
</tbody>
</table>

4.48.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.
### Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Single-Node Scenario&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>Installing Boost</td>
<td>For details, see <a href="#">4.48.4.1 Installing Boost</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Installing Samtools</td>
<td>For details, see <a href="#">Porting SAMtools</a>.</td>
</tr>
<tr>
<td>4</td>
<td>Installing bcftools</td>
<td>For details, see <a href="#">bcftools 1.10.2 Porting Guide (CentOS 7.6)</a>.</td>
</tr>
<tr>
<td>5</td>
<td>Installing BWA</td>
<td>For details, see <a href="#">BWA 0.7.17 Porting Guide (CentOS 7.6)</a>.</td>
</tr>
</tbody>
</table>

#### 4.48.4.1 Installing Boost

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Use an SFTP tool to upload the `boost_1_66_0.tar.gz` installation package to the `/path/to/DELLY` directory on the server.

**Step 3** Run the following command to decompress the installation package and go to the directory generated after the decompression:

```bash
cd /path/to/DELLY && tar -zxf boost_1_66_0.tar.gz && cd boost_1_66_0
```

**Step 4** Run the following command to perform configuration:

```bash
./bootstrap.sh --prefix=/path/to/DELLY/boost
```

**Step 5** Run the following command to perform compilation and installation:

```bash
./b2 -j16 && ./b2 install
```

----End

#### 4.48.5 Obtaining Source Code

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Download the Delly installation package and the htslib installation package and save them to the `/path/to/DELLY` directory.

Download links:
Delly installation package: https://github.com/dellytools/delly/archive/v0.8.5.tar.gz

htslib installation package: https://github.com/samtools/htslib/archive/1.10.2.tar.gz

----End

4.48.6 Compilation and Installation

Procedure

**Step 1** Run the following command to clear the environment:

```
yum erase `yum list installed |grep boost|awk {'print $1'}` -y
```

**Step 2** Run the following command to install dependencies:

```
yum install -y libcurl libcurl-devel openssl openssl-devel zlib-devel bzip2 bzip2-devel xz-devel
```

**Step 3** Run the following commands to set environment variables:

```
export CPLUS_INCLUDE_PATH=$CPLUS_INCLUDE_PATH:/path/to/boost/include
export LIBRARY_PATH=$LIBRARY_PATH:/path/to/boost/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/boost/lib
```

**Step 4** Run the following command to decompress the Delly and htslib installation packages:

```
cd /path/to/DELLY && tar -zxf v0.8.5.tar.gz && tar -zxf 1.10.2.tar.gz
```

**Step 5** Run the following command to move the htslib source code to the src/htslib directory of the Delly source code:

```
mv htslib-1.10.2/* delly-0.8.5/src/htslib/
```

**Step 6** Run the following commands to modify the Makefile file:

```
cd /path/to/DELLY/delly-0.8.5
sed -i 's/g++/`which g++`/g' Makefile
sed -i 's/\./configure/CC=`which gcc` CXX=`which g++` \./configure/g' Makefile
sed -i 's/O3/O3 -std=c++11 -march=armv8.2-a -mtune=tsv110/g' Makefile
```

**Step 7** Run the following commands to perform compilation:

```
CC=`which gcc` CXX=`which g++` make PARALLEL=1 all

**NOTE**

PARALLEL=1 indicates that the multi-thread mode is enabled. If this parameter is removed, the multi-thread mode is enabled.

**Step 8** Run the following command to load environment variables:
export PATH=/path/to/DELLY/delly-0.8.5/src:$PATH

---End

4.48.7 Running and Verifying Delly

Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to go to the test case directory:

```bash
cd /path/to/DELLY/CASE
```

**Step 3** Run the following command to decompress the test case package:

```bash
gzip -d B17NC_R1.fq.gz
```

**Step 4** Run the following command to create an index file:

```bash
bwa index -a bwtsw hg19.fa
```

**Step 5** Run the following command to create a .sam file:

```bash
bwa mem -t 96 hg19.fa B17NC_R1.fastq > bwa.sam
```

**Step 6** Run the following commands to create a .bam file and process it:

```bash
samtools view -bS bwa.sam > B17NC_R1.bam
samtools sort -@96 B17NC_R1.bam -o B17NC_R1.sort.bam
samtools index -@96 B17NC_R1.sort.bam B17NC_R1.sort.bam.bai
```

**Step 7** Run the following script to copy the .bam file to simulate the running of multiple samples at the same time:

```bash
#!/bin/bash
for varible1 in {1..127}
done
cp B17NC_R1.sort.bam B17NC_R1${varible1}.sort.bam && cp B17NC_R1.sort.bam.bai B17NC_R1${varible1}.sort.bam.bai
```

**Step 8** Run the following commands to run the test case:

```bash
cp /path/to/DELLY/delly-0.8.5/excludeTemplates/human.hg19.excl.tsv ./
{ time OMP_NUM_THREADS=128 delly call -x human.hg19.excl.tsv -o delly.bcf -g hg19.fa B17NC_R1*.sort.bam; } 2>&1 | tee -a kp920-7265.log
```

**NOTE**

Delly performs parallelization at the sample level. Therefore, **OMP_NUM_THREADS** must be equal to or less than the number of samples. In this example, there are 128 samples and **OMP_NUM_THREADS** is 128.

If the information shown in **Figure 4-26** is displayed, the running is successful.
Step 9  Run the following command to convert the running result into the VCF format and view the command output:

`bcftools view delly.bcf > delly.vcf` && `cat delly.vcf | more`

Figure 7-2  shows the command output:

---End

4.49 FastTree 2.1.11 Porting Guide (CentOS 7.6)
4.49.1 Introduction

FastTree infers approximately-maximum-likelihood phylogenetic trees from alignments of nucleotide or protein sequences.

Programming language: C

Brief description: infers approximately-maximum-likelihood phylogenetic trees.

4.49.2 Environment Requirements

**Hardware Requirements**

Table 4-242 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

**Software Requirements**

Table 4-243 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>FastTree</td>
<td>2.1.11</td>
<td>[<a href="http://meta.microbesonline.org/fasttree/">http://meta.microbesonline.org/fasttree/</a>](<a href="http://meta.microbesonline.org/fasttree/">http://meta.microbesonline.org/fasttree/</a> FastTree.c)</td>
</tr>
<tr>
<td>Test case</td>
<td>all.masked</td>
<td><a href="http://morgannprice.org/16S/403K/all.masked.gz">http://morgannprice.org/16S/403K/all.masked.gz</a></td>
</tr>
</tbody>
</table>

**OS Requirements**

Table 4-244 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
4.49.3 Paths for Software Porting

This chapter lists the software installation paths involved in the FastTree software porting.

Table 4-245 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each installation package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/FastTree</td>
<td>Installation path of FastTree.</td>
<td>The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/CASE</td>
<td>Path for storing test cases.</td>
<td></td>
</tr>
</tbody>
</table>

4.49.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

Table 4-246 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing GCC</td>
<td>For details, see 4.49.4.1 Installing GCC.</td>
</tr>
</tbody>
</table>
4.49.4.1 Installing GCC

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to install GCC:

```
yum install -y gcc*
```

----End

4.49.5 Obtaining Source Code

Procedure

Step 1 Download the FastTree C code FastTree.c from the following link:

```
http://meta.microbesonline.org/fasttree/FastTree.c
```

Step 2 Use an SFTP tool to upload the downloaded installation package to the /path/to/FastTree directory on the server.

----End

4.49.6 Compilation and Installation

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to go to the installation directory:

```
cd /path/to/FastTree
```

Step 3 Run the following command to compile the concurrent versions:

```
gcc -DOPENMP -DUSE_DOUBLE -fopenmp -O3 -march=armv8.2-a -mtune=tsv110 -finline-functions -funroll-loops -Wall -o FastTreePar-2.1.11 FastTree.c -lm
```

Step 4 Run the following command to set the maximum number of concurrent threads to 128:

```
export OMP_NUM_THREADS=128
```

Step 5 Run the following commands to add the path to the .bashrc file:

```
echo "export PATH=/path/to/FastTree:$PATH" >> /root/.bashrc
source /root/.bashrc
```

----End
4.49.7 Running and Verifying FastTree

Procedure

**Step 1**  Use PuTTY to log in to the server as the root user.

**Step 2**  Run the following command to go to the case directory and decompress the case:

```
cd /home/CASE && gzip -d all.masked.gz
```

**Step 3**  Run the following command to run the case:

```
time OMP_NUM_THREADS=128 FastTreePar-2.1.11 -fastest -noml -gtr -nt all.masked > `hostname`_`date '+%Y%m%d%H%M%S'`.masked.ft.rooted
```

*Figure 4-28* shows the command output.

```
FastTree Version 2.1.11 Double precision (No SSE3), OpenMP (128 threads)
Alignments: all.masked
Nucleotide distances: Jukes-Cantor Joints: balanced Support: Local boot 1000
Search: Fastest+2nd +NNI +SPR (2 rounds range 10) (no ML-NNI)
Topology: 1.00%bloN close->deflnl, refinal=9.59
Warning! Found k-m character(s). These are treated as gaps
Ignored unknown character B (seen 161 times)ces
Ignored unknown character O (seen 167 times)
Ignored unknown character N (seen 179 times)
Ignored unknown character X (seen 3357 times)
Ignored unknown character K (seen 2578 times)
Ignored unknown character A (seen 5590 times)
Ignored unknown character Z (seen 1239 times)
Ignored unknown character V (seen 220 times)
Ignored unknown character W (seen 299 times)
Ignored unknown character Y (seen 5404 times)
Ignored unknown character X (seen 4513 times)
Initial topology ln 2757.23 seconds 337015 01 of 337015 sets 7000)R
Refining topology: 73 rounds ML-NNI, 1 rounds ML-SPR, 0 rounds ML-NNI
Total branch-length 26.33% after 6785.91 sec of 337016 splits, 144 changes (max delta 0.000)
Total time: 10893.27 seconds Unique: 337018/337017 Bad splits: 712/337018 Worst delta-Len 0.002

cal 1.76m22s
time 827m0.97s
sys 42m17.04s
```

----End

4.49.8 More Resources

To obtain more resources, visit the official FastTree website:

[http://www.microbesonline.org/fasttree/#BranchLen](http://www.microbesonline.org/fasttree/#BranchLen)

4.50 Jellyfish 2.2.10 Porting Guide (Centos 7.6)

4.50.1 Introduction

Jellyfish is a tool for fast, memory-efficient, and multithreaded counting of k-mers in DNA. It stores k-mer counts in hash tables.

For more information Jellyfish, visit [https://github.com/gmarcais/Jellyfish](https://github.com/gmarcais/Jellyfish).

Programming language: C++
One-sentence description: k-mer counting and statistics
Open-source license: GNU

Recommended Version

The recommended version is Jellyfish-2.2.10.

4.50.2 Environment Requirements

Hardware Requirements

Table 4-247 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-248 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jellyfish</td>
<td>2.2.10</td>
<td><a href="https://github.com/gmarcais/Jellyfish/releases/download/v2.2.10/jellyfish-2.2.10.tar.gz">https://github.com/gmarcais/Jellyfish/releases/download/v2.2.10/jellyfish-2.2.10.tar.gz</a></td>
</tr>
<tr>
<td>Test case</td>
<td>human_g1k_v37</td>
<td><a href="http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz">http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-249 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

Kunpeng BoostKit for HPC

Porting Guide

Issue 14 (2021-07-23) Copyright © Huawei Technologies Co., Ltd.
4.50.3 Planning Paths for Software Porting

This section describes the software installation paths involved in the Jellyfish software porting.

Table 4-250 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/JELLYFISH</td>
<td>Installation path of Jellyfish.</td>
<td>The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/TESTCASE</td>
<td>Path for storing test cases.</td>
<td></td>
</tr>
</tbody>
</table>

4.50.4 Configuring the Compilation Environment

**Prerequisites**

Installation packages are uploaded to the server using an SFTP tool.

**Configuration Process**

Table 4-251 Configuration process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Basic environment setup</td>
<td>For details, see section &quot;Environment Setup in Cluster Scenarios&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>
4.50.5 Obtaining the Source Code

Procedure

Step 1  Download Jellyfish installation package jellyfish-2.2.10.tar.gz from the following address:
https://github.com/gmarcais/Jellyfish/releases/download/v2.2.10/jellyfish-2.2.10.tar.gz

Step 2  Download test case file human_g1k_v37.fasta from the following address:
http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz

Step 3  Use the SFTP tool to upload the Jellyfish installation package to the /path/to/JELLYFISH directory on the server.

Step 4  Use the SFTP tool to upload the human_g1k_v37.fasta test case to the /path/to/TESTCASE directory on the server.

-----End

4.50.6 Compiling and Installing Jellyfish

4.50.6.1 Installing Jellyfish

Procedure

Step 1  Run the following command to decompress the package:

tar zxvf jellyfish-2.2.10.tar.gz

Step 2  Run the following command to switch to the directory generated after decompression:

cd jellyfish-2.2.10

Step 3  Run the following command to perform configuration:

./configure --prefix=/path/to/JELLYFISH/Install CXXFLAGS='-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110 -flto' CFLAGS='-O3 -march=armv8.2-a -mtune=tsv110 -flto'

Step 4  Run the following commands to perform installation and compilation:

make -j 32
make install

Step 5  Run the following command to add the environment variable:

export PATH=/path/to/JELLYFISH/Install/bin:$PATH

-----End
4.50.7 Running and Verifying Jellyfish

Procedure

Step 1  Run the following command to switch to the directory where the test case is stored:

```bash
cd /path/to/TESTCASE
```

Step 2  Run the following command to decompress the test case file:

```bash
gzip -d human_g1k_v37.fasta.gz
```

Step 3  Run the following command to count k-mers:

```bash
jellyfish count -m 21 -s 100M -t 96 -o mer_counts -c 7 human_g1k_v37.fasta
```
If the `mer_counts` file is generated, the verification is successful.

----End

4.51 PLINK 1.9 Porting Guide (CentOS 7.6)

4.51.1 Introduction

PLINK is a free, commonly used, open-source whole-genome association analysis toolset designed by Shaun Purcell. The software is designed flexibly to perform a wide range of basic, large-scale genetic analyses.

For more information about PLINK, visit [http://www.cog-genomics.org/plink2/](http://www.cog-genomics.org/plink2/).

One-sentence description: whole-genome association analysis

Programming language: C/C++

Recommended Version

The recommended version is PLINK 1.9.

4.51.2 Environment Requirements

Hardware Requirements

**Table 4-252** lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>
**Software Requirements**

Table 4-253 lists the software requirements.

Table 4-253 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLINK</td>
<td>1.9</td>
<td><a href="https://github.com/chrchang/plink-ng/archive/master.tar.gz">https://github.com/chrchang/plink-ng/archive/master.tar.gz</a></td>
</tr>
<tr>
<td>zlib</td>
<td>1.2.11</td>
<td><a href="http://zlib.net/zlib-1.2.11.tar.gz">http://zlib.net/zlib-1.2.11.tar.gz</a></td>
</tr>
<tr>
<td>Test case</td>
<td>hapmap_r23a</td>
<td><a href="http://zzz.bwh.harvard.edu/plink/dist/hapmap_r23a.zip">http://zzz.bwh.harvard.edu/plink/dist/hapmap_r23a.zip</a></td>
</tr>
</tbody>
</table>

**OS Requirements**

Table 4-254 lists the OS requirements.

Table 4-254 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

**4.51.3 Planning Paths for Software Porting**

This section describes the software installation paths involved in the PLINK software porting.

Table 4-255 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/PLINK</td>
<td>Installation path of PLINK.</td>
<td>The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this</td>
</tr>
</tbody>
</table>
### 4.51.4 Obtaining the Source Code

**Procedure**

**Step 1** Download PLINK installation package `master.tar.gz` from the following address:

https://github.com/chrchang/plink-ng/archive/master.tar.gz

**Step 2** Download zlib installation package `zlib-1.2.11.tar.gz` from the following address:

http://zlib.net/zlib-1.2.11.tar.gz

**Step 3** Use the SFTP tool to upload the PLINK and zlib installation packages to the `/path/to/PLINK` directory on a server.

----End

### 4.51.5 Configuring the Compilation Environment

**Prerequisites**

Installation packages are uploaded to the server using an SFTP tool.

**Configuration Process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Basic environment setup</td>
<td>For details, see &quot;Environment Setup in Cluster Scenarios&quot; in the <em>HPC Solution Basic Environment Setup Guide</em>.</td>
</tr>
</tbody>
</table>

### 4.51.6 Compiling and Installing PLINK

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to install the dependency software:

```
yum install blas-devel lapack-devel atlas-devel perl-Digest-SHA -y
```
Step 3 Run the following commands to create library file links:

```
ln -s /usr/lib64/atlas/libtatlas.so.3.10 /usr/lib64/atlas/libatlas.so
ln -s /usr/lib64/atlas/libsatlas.so.3.10 /usr/lib64/atlas/libcblas.so
```

Step 4 Run the following command to go to the planned PLINK installation directory and decompress the source code package:

```
cd /path/to/PLINK && tar -zxf master.tar.gz
```

Step 5 Run the following command to move the zlib source code package to the specified directory:

```
mv zlib-1.2.11.tar.gz plink-ng-master
```

Step 6 Run the following command to switch to the installation directory and modify the configuration file:

```
cd plink-ng-master/1.9 && sed -i '0,/curl/s/curl/# curl/g' plink_first_compile
```

Step 7 Run the following command to switch to the directory generated after decompression:

```
./plink_first_compile
```

Step 8 Run the following command to configure the environment variable:

```
export PATH=/path/to/PLINK/plink-ng-master/1.9:$PATH
```

4.51.7 More Resources

To obtain more resources, visit the official website of PLINK:

```
http://zzz.bwh.harvard.edu/plink/ld.shtml
```

4.52 SNAP 2013-11-29 Porting Guide (CentOS 7.6)

4.52.1 Introduction

SNAP is a general purpose gene finding program suitable for both eukaryotic and prokaryotic genomes. SNAP is an acronym for Semi-HMM-based Nucleic Acid Parser.

For more information about SNAP, visit https://github.com/KorfLab/SNAP.

Programming language: C/C++

One-sentence description: gene sequencing

Open-source license: GNU

Recommended Version

The recommended version is SNAP-2013-11-29.
4.52.2 Environment Requirements

Hardware Requirements

Table 4-257 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-258 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cufflinks</td>
<td>2.2.1</td>
<td><a href="https://codeload.github.com/cole-trapnell-lab/cufflinks/zip/master">https://codeload.github.com/cole-trapnell-lab/cufflinks/zip/master</a></td>
</tr>
<tr>
<td>MAKER</td>
<td>3.01.03</td>
<td><a href="http://yandell.topaz.genetics.utah.edu/cgi-bin/maker_license.cgi">http://yandell.topaz.genetics.utah.edu/cgi-bin/maker_license.cgi</a></td>
</tr>
<tr>
<td>Test case 1</td>
<td>hg38.gtf</td>
<td><a href="http://genome.ucsc.edu/cgi-bin/hgTables">http://genome.ucsc.edu/cgi-bin/hgTables</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-259 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>
4.52.3 Paths for Software Porting

This section describes the software installation paths involved in the SNAP software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/CUFFLINKS</td>
<td>Installation path of Cufflinks.</td>
<td>For details, see the Cufflinks 2.2.1 Porting Guide (CentOS 7.6).</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/SNAP</td>
<td>Installation path of SNAP.</td>
<td>The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>4</td>
<td>/path/to/MAKER</td>
<td>Installation path of MAKER.</td>
<td></td>
</tr>
</tbody>
</table>

4.52.4 Configuring the Compilation Environment

Prerequisites

Installation packages are uploaded to a server using an SFTP tool.

Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Basic environment setup</td>
<td>For details, see &quot;Environment Setup in Cluster Scenarios&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Cufflinks installation</td>
<td>For details, see the Cufflinks 2.2.1 Porting Guide (CentOS 7.6).</td>
</tr>
<tr>
<td>No.</td>
<td>Configuration Item</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>MAKER installation</td>
<td>For details, see 4.52.4.1 Installing MAKER.</td>
</tr>
</tbody>
</table>

### 4.52.4.1 Installing MAKER

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to install the dependency package:

```
tar -xvf maker-3.01.03.tgz
```

**Step 3** Run the following command to set the environment variable:

```
export PATH=/path/to/MAKER/maker/src/bin:$PATH
```

--- End

### 4.52.5 Obtaining the Source Code

**Procedure**

**Step 1** Download SNAP installation package `snap-2013-11-29.tar.gz` from the following address:

```
```

**Step 2** Use the SFTP tool to upload the SNAP installation package to the `/path/to/SNAP` directory on the server.

--- End

### 4.52.6 Compiling and Installing SNAP

### 4.52.6.1 Installing SNAP

**Procedure**

**Step 1** Run the following command to decompress the package:

```
tar zxvf snap-2013-11-29.tar.gz
```

**Step 2** Run the following command to switch to the directory generated after decompression:

```
   cd snap
```

**Step 3** Run the following command to modify the Makefile file:

```
   vi Makefile
```
Press i to enter the editing mode and delete the -Werror option from line 97 to line 101.

Before the modification:

```
make $(APP)  CC="gcc" CFLAGS="-O2 -Wall -Werror"
make $(APP2) CC="gcc" CFLAGS="-O2 -Wall -Werror 
make $(APP3) CC="gcc" CFLAGS="-O2 -Wall -Werror 
make $(APP4) CC="gcc" CFLAGS="-O2 -Wall -Werror 
make $(APP5) CC="gcc" CFLAGS="-O2 -Wall -Werror 
```

After the modification:

```
make $(APP)  CC="gcc" CFLAGS="-O2 -Wall"
make $(APP2) CC="gcc" CFLAGS="-O2 -Wall"
make $(APP3) CC="gcc" CFLAGS="-O2 -Wall"
make $(APP4) CC="gcc" CFLAGS="-O2 -Wall"
make $(APP5) CC="gcc" CFLAGS="-O2 -Wall"
```

Press Esc, enter :wq!, and press Enter to save the settings and exit.

**Step 4** Run the following command to switch to the Zoe directory:

```
cd Zoe
```

**Step 5** Run the following command to modify the Makefile file:

```
vi Makefile
```

Press i to enter the editing mode and delete the -Werror option from line 94 to line 102.

Before the modification:

```
gcc:
  make $(APP)  CC="gcc" CFLAGS="-O2 -Wall -Werror"
all:
  make $(APP) CC="gcc" CFLAGS="-O2 -Wall -Werror 
  make $(APP2) CC="gcc" CFLAGS="-O2 -Wall -Werror 
  make $(APP3) CC="gcc" CFLAGS="-O2 -Wall -Werror 
  make $(APP4) CC="gcc" CFLAGS="-O2 -Wall -Werror 
  make $(APP5) CC="gcc" CFLAGS="-O2 -Wall -Werror 
```

After the modification:

```
gcc:
  make $(APP)  CC="gcc" CFLAGS="-O2 -Wall"
all:
  make $(APP) CC="gcc" CFLAGS="-O2 -Wall"
  make $(APP2) CC="gcc" CFLAGS="-O2 -Wall"
  make $(APP3) CC="gcc" CFLAGS="-O2 -Wall"
  make $(APP4) CC="gcc" CFLAGS="-O2 -Wall"
  make $(APP5) CC="gcc" CFLAGS="-O2 -Wall"
```

**Step 6** Run the following commands to perform installation and compilation:

```
cd /path/to/SNAP/snap
make
```

**Step 7** Run the following command to add the environment variable:

```
export PATH=/path/to/SNAP/snap:SPATH
```

-----End
4.52.7 Running and Verifying SNAP

Procedure

Step 1  Run the following command to declare the Cufflinks environment:
   
   ```bash
   export PATH=/path/to/CUFFLINKS/build/bin:$PATH
   ```

Step 2  Run the following command to process the gtf test case:
   
   ```bash
   gffread hg38.gtf -o- >hg38.gff3
   ```

Step 3  Run the following command to use MAKER to process the gff3 test case:
   
   ```bash
   /path/to/MAKER/maker/src/bin/maker2zff hg38.gff3
   
   The genome.ann(ZFF) and genome.dna files are generated in the current directory.
   ```

Step 4  Run the following commands to use fathom and forge to process the test cases:
   
   ```bash
   fathom genome.ann genome.dna -categorize 1000
   fathom uni.ann uni.dna -export 1000 -plus
   forge export.ann export.dna
   hmm-assembler.pl test . >test.hmm
   ```

Step 5  Run the following commands to decompress and rename the test cases:
   
   ```bash
   gzip -d GCA_000001405.15_GRCh38_full_analysis_set.fna.gz
   mv GCA_000001405.15_GRCh38_full_analysis_set.fna hs38DH.fasta
   ```

Step 6  Run the following command to perform the testing:
   
   ```bash
   { time snap test.hmm hs38DH.fasta>snap.zff ; } 2>&1 |tee snap.log
   ```

If the following information is displayed, the verification is successful:

```bash
Kunpeng BoostKit for HPC
Porting Guide
4 Others
```
4.52.8 More Resources

- **hg38.gtf** download address: [http://genome.ucsc.edu/cgi-bin/hgTables](http://genome.ucsc.edu/cgi-bin/hgTables)
- Download method: Set **output format** and **output file** as shown in the following figure, and click **get output**.

```
Table Browser

Use this program to retrieve the data associated with a track in text format, to calculate intersections between for a description of the controls in this form, and the User's Guide for general information and sample queries of your set through annotation enrichments, send the data to GREAT. Refer to the Credits page for the list of Sequence and Annotation Downloads page.
```

4.53 kmersGWAS 0.2 Porting Guide (CentOS 7.6)
4.53.1 Introduction

kmersGWAS is a library for running k-mers-based GWAS. Structural variability and polymorphism presence or non-presence are common in plant genomes.

For more information, visit the official website.

Programming language: C++, Python, and R

Brief description: a library for running k-mers-based GWAS

Recommended Version

kmersGWAS 0.2-beta

4.53.2 Environment Requirements

Hardware Requirements

Table 4-262 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-263 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEMMA</td>
<td>0.96</td>
<td><a href="https://github.com/genetics-statistics/GEMMA/archive/v0.96.tar.gz">https://github.com/genetics-statistics/GEMMA/archive/v0.96.tar.gz</a></td>
</tr>
<tr>
<td>cmake</td>
<td>3.17.3</td>
<td><a href="https://cmake.org/files/v3.17/cmake-3.17.3.tar.gz">https://cmake.org/files/v3.17/cmake-3.17.3.tar.gz</a></td>
</tr>
<tr>
<td>OpenBLAS</td>
<td>0.3.9</td>
<td><a href="https://github.com/xianyi/OpenBLAS/archive/v0.3.9.tar.gz">https://github.com/xianyi/OpenBLAS/archive/v0.3.9.tar.gz</a></td>
</tr>
<tr>
<td>eigen</td>
<td>3.3.8</td>
<td><a href="https://gitlab.com/libeigen/eigen/-/archive/3.3.8/eigen-3.3.8.tar.gz">https://gitlab.com/libeigen/eigen/-/archive/3.3.8/eigen-3.3.8.tar.gz</a></td>
</tr>
<tr>
<td>gsl</td>
<td>2.5.0</td>
<td><a href="https://github.com/ampl/gsl/archive/v2.5.0.tar.gz">https://github.com/ampl/gsl/archive/v2.5.0.tar.gz</a></td>
</tr>
<tr>
<td>Item</td>
<td>Version</td>
<td>Download URL</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>kmersGWA S</td>
<td>0.2-beta</td>
<td><a href="https://github.com/voichek/kmersGWAS/archive/v0.2-beta.tar.gz">https://github.com/voichek/kmersGWAS/archive/v0.2-beta.tar.gz</a></td>
</tr>
<tr>
<td>CTPL</td>
<td>0.0.2</td>
<td><a href="https://github.com/vit-vit/CTPL/archive/v0.0.2.tar.gz">https://github.com/vit-vit/CTPL/archive/v0.0.2.tar.gz</a></td>
</tr>
<tr>
<td>KMC</td>
<td>3.1.2rc1</td>
<td><a href="https://github.com/refresh-bio/KMC/archive/v3.1.2rc1.tar.gz">https://github.com/refresh-bio/KMC/archive/v3.1.2rc1.tar.gz</a></td>
</tr>
<tr>
<td>cxxopts</td>
<td>2.2.1</td>
<td><a href="https://github.com/jarro2783/cxxopts/archive/v2.2.1.tar.gz">https://github.com/jarro2783/cxxopts/archive/v2.2.1.tar.gz</a></td>
</tr>
<tr>
<td>Test case</td>
<td>1001G_kmers_table</td>
<td><a href="https://zenodo.org/record/3596255/files/A_thaliana_1001G_kmers_table.tar.gz">https://zenodo.org/record/3596255/files/A_thaliana_1001G_kmers_table.tar.gz</a></td>
</tr>
</tbody>
</table>

**OS Requirements**

Table 4-264 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
</tbody>
</table>

**4.53.3 Planning the Paths for Software Porting**

This section describes the software installation paths involved in the kmersGWAS software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/R</td>
<td>Installation path of R.</td>
<td>The installation paths listed in this table are only for reference. Shared paths are recommended. Replace the paths used in commands in this document with the actual</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/OPENBLAS</td>
<td>Installation path of OpenBLAS.</td>
<td></td>
</tr>
</tbody>
</table>

Issue 14 (2021-07-23)  Copyright © Huawei Technologies Co., Ltd.  725
### 4.53.4 Configuring the Compilation Environment

**Prerequisites**

Installation packages are uploaded to a server using an SFTP tool.

**Configuration Process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot; in the <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>Install R.</td>
<td>For details, see <a href="#">4.53.4.1 Installing R</a>.</td>
</tr>
<tr>
<td>3</td>
<td>Install CMake.</td>
<td>For details, see <a href="#">4.53.4.2 Installing CMake</a>.</td>
</tr>
<tr>
<td>4</td>
<td>Install OpenBLAS.</td>
<td>For details, see <a href="#">4.53.4.3 Installing OpenBLAS</a>.</td>
</tr>
<tr>
<td>5</td>
<td>Install Eigen.</td>
<td>For details, see <a href="#">4.53.4.4 Installing Eigen</a>.</td>
</tr>
<tr>
<td>6</td>
<td>Install GSL</td>
<td>For details, see <a href="#">4.53.4.5 Installing GSL</a>.</td>
</tr>
<tr>
<td>7</td>
<td>Install Gemma</td>
<td>For details, see <a href="#">4.53.4.6 Installing Gemma</a>.</td>
</tr>
</tbody>
</table>
4.53.4.1 Installing R

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use the SFTP tool to upload the R installation package to the /path/to/R directory.

Step 3 Run the following command to decompress the installation package and go to the installation directory:

```
cd /path/to/R && tar -zxf R-3.1.1.tar.gz && cd R-3.1.1
```

Step 4 Run the following command to perform configuration:

```
./configure --enable-R-shlib --enable-R-static-lib --with-libpng --with-jpeglib --prefix=/path/to/R
```

Step 5 Run the following command to perform compilation and installation:

```
mak all -j8 && make install
```

----End

4.53.4.2 Installing CMake

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use the SFTP tool to upload the CMake installation package to the /path/to/CMAKE directory.

Step 3 Run the following command to decompress the installation package:

```
cd /path/to/CMAKE && tar -zxf cmake-3.17.3.tar.gz
```

Step 4 Run the following command to switch to the installation directory:

```
cd cmake-3.17.3
```

Step 5 Run the following commands to perform compilation and installation:

```
CC=`which gcc` CXX=`which g++` ./bootstrap --prefix=/path/to/CMAKE

gmake && gmake install
```

Step 6 Run the following command to set the environment variable:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

----End

4.53.4.3 Installing OpenBLAS

Procedure

Step 1 Use PuTTY to log in to the server as the root user.
Step 2 Use the SFTP tool to upload the OpenBLAS installation package to the /path/to/OPENBLAS directory.

Step 3 Run the following command to install dependencies:

```
yum install blas blas-devel lapack lapack-devel -y
```

Step 4 Run the following command to decompress the installation package:

```
cd /path/to/OPENBLAS && tar -zxf OpenBLAS-0.3.9.tar.gz
```

Step 5 Run the following command to switch to the directory generated after decompression:

```
cd OpenBLAS-0.3.9/
```

Step 6 Run the following commands to set environment variables:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 7 Run the following commands to perform compilation and installation:

```
make -j 32
make PREFIX=/path/to/OPENBLAS install
```

----End

4.53.4.4 Installing Eigen

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use the SFTP tool to upload the Eigen installation package to the /path/to/EIGEN directory.

Step 3 Run the following command to decompress the installation package:

```
cd /path/to/EIGEN && tar -zxf eigen-3.3.8.tar.gz
```

Step 4 Run the following command to switch to the directory generated after decompression:

```
cd eigen-3.3.8
```

Step 5 Run the following commands to perform compilation and installation:

```
mkdir build
cd build
make -j 32
make PREFIX=/path/to/EIGEN install
```

----End
4.53.4.5 Installing GSL

Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Use the SFTP tool to upload the GSL installation package to the `/path/to/GSL` directory.

**Step 3** Run the following command to decompress the installation package:
```
  cd /path/to/GSL && tar -zxf gsl-2.5.0.tar.gz
```

**Step 4** Run the following command to switch to the directory generated after decompression:
```
  cd gsl-2.5.0
```

**Step 5** Run the following commands to perform compilation and installation:
```
  mkdir build
  cd build
  ../configure --prefix=/path/to/GSL
  make -j 32
  make install
```

----End

4.53.4.6 Installing Gemma

Procedure

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Use the SFTP tool to upload the Gemma installation package to the `/path/to/GEMMA` directory.

**Step 3** Run the following command to install the dependent libraries:
```
  yum install blas-devel lapack-devel -y
```

**Step 4** Run the following command to decompress the installation package:
```
  cd /path/to/GEMMA && tar -zxf GEMMA-0.96.tar.gz
```

**Step 5** Run the following command to switch to the directory generated after decompression:
```
  cd GEMMA-0.96
```

**Step 6** Run the following command to modify compilation optimization parameters:
```
  sed -i 's/\-O3/\-O3 -march=armv8.2-a -mtune=tsv110/ Makefile
```

**Step 7** Run the following command to replace the parameter `-m64` by `-mabi=lp64`:
```
  sed -i 's/\-m64/\-mabi=lp64/g' Makefile
```
Step 8  Run the following commands to change the value of the compilation parameter `LIBS_LNX_S_LAPACK` and delete the compilation parameter `static`:

```
sed -i 's/LIBS_LNX_S_LAPACK = /# LIBS_LNX_S_LAPACK = /' Makefile
sed -i '/LIBS_LNX_S_LAPACK = /a LIBS_LNX_S_LAPACK = -lgfortran -lopenblas -llapack -lblas -lgfortran' Makefile
sed -i 's/CPPFLAGS += -static/#CPPFLAGS += -static/' Makefile
```

Step 9  Run the following commands to set environment variables:

```
export LIBRARY_PATH=/path/to/OPENBLAS/lib:/path/to/GSL/lib:$LIBRARY_PATH
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:/path/to/GSL/lib:
SLD_LIBRARY_PATH
export CPLUS_INCLUDE_PATH=/path/to/OpenBLAS/include:/path/to/EIGEN/include:
eigen3:/path/to/GSL/include:SCPLUS_INCLUDE_PATH
export C_INCLUDE_PATH=/path/to/OpenBLAS/include:
eigen3:/path/to/GSL/include:
```

Step 10 Run the following command to perform compilation and installation:

```
make -j
```

4.53.5 Obtaining the Source Code

Procedure

Step 1  Download the kmersGWAS, KMC, CTPL, and cxxopts source code packages.

Download the kmersGWAS source code package at [https://github.com/voichek/kmersGWAS/archive/v0.2-beta.tar.gz](https://github.com/voichek/kmersGWAS/archive/v0.2-beta.tar.gz)

Download the KMC source code package at [https://github.com/refresh-bio/KMC/archive/v3.1.2rc1.tar.gz](https://github.com/refresh-bio/KMC/archive/v3.1.2rc1.tar.gz)

Download the CTPL source code package at [https://github.com/vit-vit/CTPL/archive/v.0.0.2.tar.gz](https://github.com/vit-vit/CTPL/archive/v.0.0.2.tar.gz)

Download the cxxopts source code package at [https://github.com/jarro2783/cxxopts/archive/v2.2.1.tar.gz](https://github.com/jarro2783/cxxopts/archive/v2.2.1.tar.gz)

Step 2  Use the SFTP tool to upload the software packages to the `/path/to/KMERSGWAS` directory.

----End
4.53.6 Compiling and Installing kmersGWAS

Procedure

<i>Note</i> If kmersGWAS is installed on an x86 server, you can skip Step 2, Step 8, Step 9, and Step 10.

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following commands to add the sse2neon dependency:

```bash
cd /path/to/KMERSGWAS
wget https://github.com/DLTcollab/sse2neon/archive/master.tar.gz --no-check-certificate -O sse2neon-master.tar.gz
tar -zxf sse2neon-master.tar.gz
```

```bash
export CPLUS_INCLUDE_PATH=/path/to/KMERSGWAS/sse2neon-master:
```

```bash
export C_INCLUDE_PATH=/path/to/KMERSGWAS/sse2neon-master:
```

**Step 3** Run the following commands to decompress the installation package:

```bash
cd /path/to/KMERSGWAS
tar -zxf kmersGWAS-0.2-beta.tar.gz
```

**Step 4** Run the following commands to copy the KMC package to kmersGWAS:

```bash
tar -zxf KMC-3.1.2rc1.tar.gz
cp -rf KMC-3.1.2rc1/* kmersGWAS-0.2-beta/include/KMC/
```

**Step 5** Run the following commands to copy the CTPL package to kmersGWAS:

```bash
tar -zxf CTPL-v.0.0.2.tar.gz
cp -rf CTPL-v.0.0.2/* kmersGWAS-0.2-beta/include/CTPL/
```

**Step 6** Run the following commands to copy the cxxopts package to kmersGWAS:

```bash
tar -zxf cxxopts-2.2.1.tar.gz
cp -rf cxxopts-2.2.1/* kmersGWAS-0.2-beta/include/cxxopts/
```

**Step 7** Run the following command to switch to the kmersGWAS installation directory:

```bash
cd kmersGWAS-0.2-beta
```

**Step 8** Run the following command to delete the -msse4.2 compilation item:

```bash
sed -i 's/-msse4.2//g' Makefile
```

**Step 9** Run the following command to add compilation optimization parameters:

```bash
sed -i 's/-O3/-O3 -march=armv8.2-a -mtune=tsv110/g' Makefile
```
Step 10  Replace the nmmintrin.h reference file.

- Run the following commands to comment out nmmintrin.h and smmintrin.h:
  
  ```
  sed -i 's/#include <nmmintrin.h>///#include <nmmintrin.h>/g;s/
  #include <smmintrin.h>///#include <smmintrin.h>/g' src/
  kmers_multiple_databases.cpp
  ```

  ```
  sed -i 's/#include <smmintrin.h>///#include <smmintrin.h>/g' src/
  snps_multiple_databases.cpp
  ```

- Run the following commands to import the header file sse2neon.h:
  ```
  sed -i '/#include <nmmintrin.h>/ i
  #include "sse2neon.h"' src/
  kmers_multiple_databases.cpp
  ```

  ```
  sed -i '/#include <smmintrin.h>/ i
  #include "sse2neon.h"' src/
  snps_multiple_databases.cpp
  ```

Step 11  Run the following command to perform compilation and installation:

```
make -j
```

Step 12  Run the following commands to delete the original gemma_0_96 and create a new soft link:

```
cd /path/to/KMERSGWAS/kmersGWAS-0.2-beta/external_programs
rm -rf gemma_0_96
ln -s /path/to/GEMMA/bin/gemma gemma_0_96
```

----End

4.53.7 Running and Verifying kmersGWAS

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following commands to set environment variables:

```
export PATH=/path/to/KMERSGWAS/kmersGWAS-0.2-beta/bin:$PATH
export PATH=/path/to/R/bin:$PATH
export LD_LIBRARY_PATH=/path/to/OPENBLAS/lib:$LD_LIBRARY_PATH
```

Step 3  Run the following commands to go to the directory for storing test cases, decompress the cases, and move them to the specified directory:

```
cd /path/to/CASE

tar -zxf A_thaliana_1001G_kmers_table.tar.gz

mv A_thaliana_1001G_kmers_table /path/to/KMERSGWAS/kmersGWAS-0.2-beta/examples/flowering_time_arabidopsis/
```

Step 4  Run the following command to go to the directory of test cases:

```
cd /path/to/KMERSGWAS/kmersGWAS-0.2-beta/examples/flowering_time_arabidopsis/
```
Step 5 Run the following command to run the case:

```bash
{ time python2.7 ../../kmers_gwas.py --pheno FT10.pheno --kmers_table A_thaliana_1001G_kmers_table/kmers_table -l 31 -p 128 --outdir run_GWAS_FT10_kp_`date '+%Y%m%d%H%M%S'`; } 2>&1 | tee -a `hostname`_`date '+%Y%m%d%H%M%S'`.log
```

Figure 4-29 shows an example of the running result.

```
<table>
<thead>
<tr>
<th>pve estimate</th>
<th>se(pve)</th>
<th>Reading SNPs</th>
<th>pve estimate</th>
<th>se(pve)</th>
<th>Reading SNPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.999967</td>
<td>0.00%</td>
<td>100.06%</td>
<td>0.999967</td>
<td>0.00%</td>
<td>100.06%</td>
</tr>
</tbody>
</table>
```

real 50m7.595s
user 5521m25.976s
sys 1m6.051s

----End

4.54 PAML 4.9j Porting Guide (CentOS 7.6)

4.54.1 Introduction

PAML is a package of programs for phylogenetic analyses of DNA or protein sequences using maximum likelihood.

For more information about PAML, visit the official PAML website.

Programming language: Python

One-sentence description: phylogenetic analyses of DNA or protein sequences

Recommended Version

The recommended version is PAML4.9j.

4.54.2 Environment Requirements

Hardware Requirements

Table 4-267 lists the hardware requirements.
Table 4-267 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-268 lists the software requirements.

Table 4-268 Software requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAML</td>
<td>4.9</td>
<td><a href="http://abacus.gene.ucl.ac.uk/software/paml4.9j.tgz">http://abacus.gene.ucl.ac.uk/software/paml4.9j.tgz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-269 lists the OS requirements.

Table 4-269 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

4.54.3 Planning Paths for Software Porting

This section describes the software installation paths involved in the PAML software porting.

Table 4-270 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>No.</td>
<td>Software Installation Path</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/PAML</td>
<td>Installation path of PAML.</td>
<td>The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.</td>
</tr>
</tbody>
</table>

### 4.54.4 Obtaining the Source Code

**Procedure**

**Step 1** Download PAML source code package `paml4.9j.tgz` from the following address:

[http://abacus.gene.ucl.ac.uk/software/paml4.9j.tgz](http://abacus.gene.ucl.ac.uk/software/paml4.9j.tgz)

**Step 2** Use the SFTP tool to upload the source code package to the `/path/to/PAML/` directory on the server.

---End

### 4.54.5 Compiling and Installing PAML

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to decompress the PMAL installation package:

`tar xf paml4.9j.tgz`

**Step 3** Run the following command to switch to the directory generated after decompression:

`cd paml4.9j`

**Step 4** Run the following commands to delete the Windows executable file (.exe file) from the `paml4.9j/bin/` folder:

`cd /path/to/PAML/paml4.9j/bin`

`rm *.exe`

**Step 5** Run the following commands to perform compilation:

`cd /path/to/PAML/paml4.9j/src`

`make -f Makefile`

`ls -lf`
Step 6 Run the following commands to copy the executable file in the src directory to the bin directory:

```
cp /path/to/PAML/paml4.9j/src/baseml basemlg codeml pamp evolver yn00 chi2 /path/to/PAML/paml4.9j/bin
```

cd ..

```
lsof -F bin
```

If the following three files exist in the bin directory, the compilation is successful:

```
bin/baseml
bin/codeml
bin/evolver
```

4.54.6 Running and Verifying PAML

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to switch to the installation directory of the executable file:

```
cd /path/to/PAML/paml4.9j
```

Step 3 Run the following commands to run the test case:

```
cp src/codeml /path/to/PAML/paml4.9j/examples/HIVNSsites
cd /path/to/PAML/paml4.9j/examples/HIVNSsites
./codeml
```

After the command is run, the following information is displayed:

```
53 h-m-p 1.4141 0.0000 0.0000 C 1106.445011 0.03024 1833 | 0/28
54 h-m-p 0.0000 0.0000 0.0000 C 1106.445005 0.07664 1833 | 0/28
55 h-m-p 1.0000 0.0000 0.0000 Y 1106.445004 0.07664 1053 | 0/28
56 h-m-p 1.0000 0.0000 0.0000 Y 1106.445004 0.31362 2019 | 0/28
57 h-m-p 1.0000 0.0000 0.0000 Y 1106.445004 0.21154 2069 | 0/28
58 h-m-p 1.0000 0.0000 0.0000 C 1106.445004 0.10300 2129
Out...
```

4.54.7 More Resources

To obtain more resources, visit the abacus page of PAML:

```
http://abacus.gene.ucl.ac.uk/software/paml.html
```
4.55 Diamond 2.0.4 Porting Guide (CentOS 7.6)

4.55.1 Introduction

Diamond is a sequence aligner for protein and translated DNA searches, designed for high performance analysis of big sequence data. The key features are:

For more information about Diamond, visit the official Diamond website.

Programming language: C++

Brief description: a sequence aligner for protein and translated DNA searches

Open-source license: GPL3.0

Recommended Version
diamond 2.0.4

4.55.2 Environment Requirements

Hardware Requirements

Table 4-271 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 2-2 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>diamond</td>
<td>2.0.4</td>
<td><a href="https://github.com/bbuchfink/diamond/archive/v2.0.4.tar.gz">https://github.com/bbuchfink/diamond/archive/v2.0.4.tar.gz</a></td>
</tr>
<tr>
<td>cmake</td>
<td>3.17.3</td>
<td><a href="https://cmake.org/files/v3.17/cmake-3.17.3.tar.gz">https://cmake.org/files/v3.17/cmake-3.17.3.tar.gz</a></td>
</tr>
<tr>
<td>Item</td>
<td>Version</td>
<td>Download Link</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Test case 2</td>
<td>human_g1k_v37</td>
<td><a href="http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz">ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz</a></td>
</tr>
</tbody>
</table>

**OS Requirements**

Table 4-273 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

**4.55.3 Paths for Software Porting**

This chapter lists the software installation paths involved in the Diamond software porting.

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each installation package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/DIAMOND</td>
<td>Installation path of Diamond.</td>
<td>The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/TESTCASE</td>
<td>Path for storing test cases.</td>
<td></td>
</tr>
</tbody>
</table>
4.55.4 Configuring the Compilation Environment

Prerequisites

The installation packages have been uploaded to the server using an SFTP tool.

Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Setting Up the Basic Environment</td>
<td>For details, see &quot;Setting Up the Environment for the Cluster Scenario&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Installing CMake</td>
<td>For details, see 4.55.4.1 Installing CMake.</td>
</tr>
</tbody>
</table>

4.55.4.1 Installing CMake

Procedure

Step 1  
Use PuTTY to log in to the server as the root user.

Step 2  
Run the following command to decompress the CMake installation package:

tar zxvf cmake-3.17.3.tar.gz

Step 3  
Run the following command to go to the directory generated after decompression:

cd cmake-3.17.3

Step 4  
Run the following command to perform configuration:

./configure --prefix=/path/to/CMAKE

Step 5  
Run the following commands to perform compilation and installation:

make
make install

Step 6  
Run the following command to set CMake environment variables:

export PATH=/path/to/CMAKE/bin:$PATH

-----End
4.55.5 Obtaining Source Code

Procedure

Step 1  Download the Diamond source code package diamond-2.0.4.tar.gz from the following link:

https://github.com/bbuchfink/diamond/archive/v2.0.4.tar.gz

Step 2  Use an SFTP tool to upload the downloaded installation package to the /path/to/diamond directory on the server.

----End

4.55.6 Compilation and Installation

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Run the following command to decompress the Diamond installation package:

tar -xvf diamond-2.0.4.tar.gz

Step 3  Run the following command to go to the directory generated after decompression:

    cd diamond-2.0.4

Step 4  Run the following commands to create a folder:

    mkdir build

Step 5  Run the following commands to perform compilation and installation:

    cd build
    export CC=`which gcc`
    export CXX=`which g++`
    export FC=`which gfortran`
    cmake -DCMAKE_INSTALL_PREFIX=/path/to/DIAMOND ../
    make -j 16
    make install

Step 6  Run the following command to set environment variables:

    export PATH=/path/to/DIAMOND/bin:SPATH

----End
4.55.7 Running and Verifying Diamond

**Procedure**

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Run the following command to go to the case directory and decompress the case:

```
cd /path/to/TESTCASE

gzip -d human_g1k_v37.fasta.gz

gzip -d swissprot.gz
```

**Step 3** Run the following command to run the case:

```
diamond blastp -p 128 -q human_g1k_v37.fasta -d swissprot -o out.tsv --very-sensitive
```

After the command is executed, the following information is displayed:

```
.......
Loading reference sequences... [0s]
Deallocating buffers... [0s]
Deallocating queries... [0.002s]
Loading query sequences... [0s]
Closing the input file... [0s]
Closing the output file... [0s]
Closing the database file... [0.003s]
Deallocating taxonomy... [0s]
Total time = 538.485s
Reported 0 pairwise alignments, 0 HSPs.
0 queries aligned.
```

----End

4.55.8 More Resources

For more information, visit the official Diamond website:

http://www.diamondsearch.org/

4.56 HISAT 0.1.6 Porting Guide (CentOS 7.6)

4.56.1 Introduction

HISAT is a fast and sensitive spliced alignment program for mapping RNA-seq reads. It includes a new indexing scheme based on the Burrows-Wheeler transform (BWT) and the FM index, called hierarchical indexing, that employs the two types of indexes: one global FM index representing the whole genome, and many separate local FM indexes for small regions collectively covering the genome. All these together, HISAT enables extremely fast and sensitive alignment of reads, in particular those spanning two exons or more.

For more information about HISAT, visit the official HISAT website.

Programming languages: C/C++
Brief description: a fast and sensitive spliced alignment program for mapping RNA-seq reads
Open-source license: GPL

**Recommended Version**

hisat-0.1.6

### 4.56.2 Environment Requirements

**Hardware Requirements**

Table 4-276 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

**Software Requirements**

Table 4-277 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>HISAT</td>
<td>0.1.6</td>
<td><a href="http://www.ccb.jhu.edu/software/hisat/downloads/hisat-0.1.6-beta-source.zip">http://www.ccb.jhu.edu/software/hisat/downloads/hisat-0.1.6-beta-source.zip</a></td>
</tr>
<tr>
<td>Test case file 1</td>
<td>R1.fq</td>
<td><a href="http://opengene.org/data/R1.fq.gz">http://opengene.org/data/R1.fq.gz</a></td>
</tr>
<tr>
<td>Test case file 2</td>
<td>R2.fq</td>
<td><a href="http://opengene.org/data/R2.fq.gz">http://opengene.org/data/R2.fq.gz</a></td>
</tr>
</tbody>
</table>

**OS Requirements**

Table 4-278 lists the OS requirements.
Table 4-278 OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.56.3 Paths for Software Porting

This chapter lists the software installation paths involved in the HISAT software porting.

Table 4-279 Paths for software porting

<table>
<thead>
<tr>
<th>N o.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each installation package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in <a href="#">HPC Solution Basic Environment Setup Guide</a>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/HISAT</td>
<td>Installation path of HISAT.</td>
<td>The installation paths are only examples. Shared paths are recommended. Replace the paths used in commands in this document with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/TESTCASE</td>
<td>Path for storing the test cases.</td>
<td></td>
</tr>
</tbody>
</table>

### 4.56.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages have been uploaded to the server using an SFTP tool.
4.56.5 Obtaining Source Code

Procedure

**Step 1** Download the HISAT installation package **hisat-0.1.6-beta-source.zip** from the following link:

http://www.ccb.jhu.edu/software/alias/downloads/hisat-0.1.6-beta-source.zip

**Step 2** Download the test case files **R1.fq.gz**, **R2.fq.gz**, and **Drosophila_melanogaster.BDGP6.dna.toplevel.fa.gz**.

Download link 1: http://opengene.org/data/R1.fq.gz

Download link 2: http://opengene.org/data/R2.fq.gz


**Step 3** Use an SFTP tool to upload the HISAT installation package to the `/path/to/HISAT` directory on the server.

**Step 4** Use an SFTP tool to upload the test cases to the `/path/to/TESTCASE` directory on the server.

----End

4.56.6 Compilation and Installation

Prerequisites

You have obtained the **sse2neon.h** file, which is available at [https://github.com/DLTcollab/sse2neon/blob/master/sse2neon.h](https://github.com/DLTcollab/sse2neon/blob/master/sse2neon.h).

Procedure

**Step 1** Run the following command to decompress the installation package:

unzip hisat-0.1.6-beta-source.zip

**Step 2** Run the following command to go to the directory generated after decompression:
cd hisat-0.1.6-beta

**Step 3** Run the following command to set environment variables:

```bash
export NO_TBB=1
export POPCNT_CAPABILITY=0
```

**Step 4** Run the following command to modify the *Makefile* file:

1. Run the following command to open the *Makefile* file:
   ```bash
   vi Makefile
   ```
2. Press `i` to go to the edit mode. Modify the aarch64 branch of `BITS=64` in the *Makefile* file. Specifically, between line 143 and line 144, insert the following content in bold: (*BITS=64* supports x86 only and does not support aarch64. You need to add an aarch64.)

   **Before the modification:**
   ```bash
   BITS=32
   ifeq (x86_64,$(shell uname -m))
   BITS=64
   endif
   ```

   **After the modification:**
   ```bash
   BITS=32
   ifeq (x86_64,$(shell uname -m))
   BITS=64
   endif
   ifeq (aarch64,$(shell uname -m))
   BITS=64
   endif
   ```
3. Press `Esc`, type `:wq!`, and then press `Enter` to save and exit.

**Step 5** Run the following commands to modify the *Makefile* file:

1. Run the following command to open the *Makefile* file:
   ```bash
   vi Makefile
   ```
2. Press `i` to go to the edit mode. Modify the aarch64 branches of `-m64` and `-msse2` in the *Makefile* file. Specifically, between line 165 and line 168, insert the following content in bold: The compilation options `-m64` and `-msse2` in *Makefile* support x86 only and do not support aarch64. You need to add aarch64 branches.

   **Before the modification:** (lines 152 to 159 in the original file)
   ```bash
   ifeq (32,$(BITS))
   BITS_FLAG = -m32
   endif
   ifeq (64,$(BITS))
   BITS_FLAG = -m64
   endif
   SSE_FLAG=-msse2
   ```

   **After the modification:**
   ```bash
   ifeq (32,$(BITS))
   BITS_FLAG = -m32
   endif
   ifeq (64,$(BITS))
   BITS_FLAG = -m64
   endif
   ifeq (aarch64,$(shell uname -m))
   BITS_FLAG =
   ```
3. Press Esc, type :wq!, and then press Enter to save and exit.

**Step 6** Run the following commands to modify the Makefile file:

1. Run the following command to open the Makefile file:
   ```bash
   vi Makefile
   ```
2. Press i to go to the edit mode and modify the FLAGS parameters in the Makefile file.

   **Before the modification:**
   ```
   DEBUG_FLAGS    = -O0 -g3 $(BIToS_FLAG) $(SSE_FLAG)
   DEBUG_DEFS     = -DCOMPILER_OPTIONS=""$(DEBUG_FLAGS) $(EXTRA_FLAGS)"
   RELEASE_FLAGS  = -O3 $(BITS_FLAG) $(SSE_FLAG) -funroll-loops -g3
   RELEASE_DEFS   = -DCOMPILER_OPTIONS=""$(RELEASE_FLAGS) $(EXTRA_FLAGS)"
   NOASSERT_FLAGS = -DNDEBUG
   FILE_FLAGS     = -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE
   ```

   **After the modification:**
   ```
   DEBUG_FLAGS    = -O0 -g3 $(BIToS_FLAG) $(SSE_FLAG) -funroll-loops -std=c++98
   DEBUG_DEFS     = -DCOMPILER_OPTIONS=""$(DEBUG_FLAGS) $(EXTRA_FLAGS)"
   RELEASE_FLAGS  = -O3 $(BITS_FLAG) $(SSE_FLAG) -funroll-loops -g3 -std=c++98 -Xlinker --allow-multiple-definition
   RELEASE_DEFS   = -DCOMPILER_OPTIONS=""$(RELEASE_FLAGS) $(EXTRA_FLAGS)"
   NOASSERT_FLAGS = -DNDEBUG
   FILE_FLAGS     = -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE
   ```

3. Press Esc, type :wq!, and then press Enter to save and exit.

**Step 7** Run the following command to copy **sse2neon.h** to the installation directory of the compiler.

   ```bash
   cp sse2neon.h /path/to/NG/\include
   ```

**Step 8** Run the following commands to modify the **aligner_sw.h** and **sse_util.h** header files:

1. Run the following command to open the **aligner_sw.h** file:
   ```bash
   vi aligner_sw.h
   ```
2. Press i to go to the edit mode. Modify the **aligner_sw.h** file by replacing
   ```
   #include <emmintrin.h>
   ```
   with
   ```
   #include <sse2neon.h>
   ```

   **Before the modification:**
   ```
   #include <emmintrin.h>
   ```

   **After the modification:**
   ```
   #include <sse2neon.h>
   ```

3. Press Esc, type :wq!, and then press Enter to save and exit.

**NOTE**

Modify the **sse_util.h** file in the same way.

**Step 9** Run the following command to perform compilation and installation:

   ```bash
   make
   ```

**Step 10** Run the following command to add environment variables:

   ```bash
   export PATH=/path/to/HISAT/hisat-0.1.6-beta:$PATH
   ```

    ----End
4.56.7 Running and Verifying HISAT

Procedure

Step 1 Run the following command to go to the test case directory:

```
cd /path/to/TESTCASE
```

Step 2 Run the following commands to decompress the test case file packages:

```
gzip -d Drosophila_melanogaster.BDGP6.dna.toplevel.fa.gz
gzip -d R1.fq.gz
gzip -d R2.fq.gz
```

Step 3 Run the following command to rename the fa file:

```
mv Drosophila_melanogaster.BDGP6.dna.toplevel.fa genome.fa
```

Step 4 Run the following command to create a genome index:

```
hisat-build genome.fa genome
```

Step 5 Run the following command to compare gene sequences:

```
hisat -t -x genome -p 96 -1 R1.fq -2 R2.fq -S result.sam
```

If the following information is displayed, the verification is successful:

```
Time loading reference: 00:08:06
Time loading forward index: 00:08:06
875000 reads; of those:
375000 (100.0%) were paired; of those:
375000 (100.0%) aligned concordantly 0 times
0 (0.0%) aligned concordantly exactly 1 time
0 (0.0%) aligned concordantly >1 times
-----
375000 pairs aligned concordantly 0 times; of those:
0 (0.0%) aligned discordantly 1 time
-----
375000 pairs aligned 0 times concordantly or discordantly; of those:
7199990 mates make up the pairs; of those:
7199990 (100.0%) aligned 0 times
1 (0.0%) aligned exactly 1 time
0 (0.0%) aligned >1 times
0.0% overall alignment rate
Time searching: 00:02:48
Overall time: 00:10:24
```

-----End

4.57 MACS2 2.1.4 Porting Guide (CentOS 7.6)

4.57.1 Introduction

With the improvement of sequencing techniques, chromatin immunoprecipitation followed by high throughput sequencing (ChIP-Seq) is getting popular to study genome-wide protein-DNA interactions. To address the lack of powerful ChIP-Seq analysis method, we presented the Model-based Analysis of ChIP-Seq (MACS), for identifying transcript factor binding sites. MACS captures the influence of genome complexity to evaluate the significance of enriched ChIP regions and MACS improves the spatial resolution of binding sites through combining the information
of both sequencing tag position and orientation. MACS can be easily used for ChIP-Seq data alone, or with a control sample with the increase of specificity.

Programming language: Python 2

Recommended Version

The recommended version is MACS2-2.1.4.

4.57.2 Environment Requirements

Hardware Requirements

Table 4-281 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-282 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACS2</td>
<td>2.1.4</td>
<td><a href="https://codeload.github.com/macs3-project/MACS/zip/v2.1.4">https://codeload.github.com/macs3-project/MACS/zip/v2.1.4</a></td>
</tr>
<tr>
<td>NumPy</td>
<td>3.17.3</td>
<td><a href="https://files.pythonhosted.org/packages/b7/6f/24647f014eeff9b67a24adfbcd4f4928349b4a0f8393b3d7fe648d4d2de3/numpy-1.16.6.zip">https://files.pythonhosted.org/packages/b7/6f/24647f014eeff9b67a24adfbcd4f4928349b4a0f8393b3d7fe648d4d2de3/numpy-1.16.6.zip</a></td>
</tr>
<tr>
<td>Cython</td>
<td>0.29.21</td>
<td><a href="https://files.pythonhosted.org/packages/6c/9f/f501ba9d178aeb1f5bf7da1ad5619b207c90ac235d9859961c11829d0160/Cython-0.29.21.tar.gz">https://files.pythonhosted.org/packages/6c/9f/f501ba9d178aeb1f5bf7da1ad5619b207c90ac235d9859961c11829d0160/Cython-0.29.21.tar.gz</a></td>
</tr>
</tbody>
</table>

OS Requirements

Table 4-283 lists the OS requirements.
### 4.57.3 Planning Paths for Software Porting

This section describes the software installation paths involved in the MACS2 software porting.

**Table 4-284 Paths for software porting**

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the <strong>HPC Solution Basic Environment Setup Guide</strong>.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/MACS2</td>
<td>Installation path of MACS2.</td>
<td>The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/ CYTHON</td>
<td>Installation path of Cython.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/ NUMPY</td>
<td>Installation path of NumPy.</td>
<td></td>
</tr>
</tbody>
</table>

### 4.57.4 Configuring the Compilation Environment

**Prerequisites**

Installation packages are uploaded to a server using an SFTP tool.

**Configuration Process**

**Table 4-285 Configuration process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>python-devel</td>
<td>For details, see <a href="#4-57-4-1-installing-python-devel">4.57.4.1 Installing python-devel</a></td>
</tr>
<tr>
<td>No.</td>
<td>Configuration Item</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>2</td>
<td>Cython</td>
<td>For details, see 4.57.4.2 Installing Cython.</td>
</tr>
<tr>
<td>3</td>
<td>numpy</td>
<td>For details, see 4.57.4.3 Installing NumPy.</td>
</tr>
</tbody>
</table>

### 4.57.4.1 Installing python-devel

**Procedure**

**Step 1** Use PuTTY to log in to the server as the *root* user.

**Step 2** Run the following command to use the Yum tool to install the basic dependency package:

```
yum install python-devel
```

----End

### 4.57.4.2 Installing Cython

**Procedure**

**Method 1**

**Step 1** Use PuTTY to log in to the server as the *root* user.

**Step 2** Run the following command to switch to the Cython directory:

```
cd /path/to/CYTHON
```

**Step 3** Run the following command to install pip for Python 2:

```
pip install cython
```
Alternatively, run the following command:

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple cython
```

----End

**Method 2**

**Step 1** Download the Cython file from the following address:

```
https://files.pythonhosted.org/packages/6c/9f/f501ba9d178aeb1f5bf7da1ad5619b207c90ac235d9859961c11829d0160/Cython-0.29.21.tar.gz
```

**Step 2** Run the following command to install Cython:

```
python setup.py install
```

**Step 3** Run the following command to load Cython using pip:
pip install Cython

4.57.4.3 Installing NumPy

Step 1 Run the following command to obtain the NumPy source code:

```bash
wget https://files.pythonhosted.org/packages/b7/6f/
24647f014eeef9b67a24adfcbcd4f4928349b4a0f8393b3d7fe648d4d2de3/
numpy-1.16.6.zip
```

Step 2 Run the following command to decompress the package:

```bash
unzip numpy-1.16.6.zip
```

Step 3 Run the following command to switch to the directory generated after decompression:

```bash
cd numpy-1.16.6
```

Step 4 Run the following command to install the program:

```bash
python setup.py install
```

4.57.5 Obtaining the Source Code

Procedure

Step 1 Download MACS2 source code package MACS2-2.1.4.zip.

You are advised to download the software package from the web page of the PC on the external network and copy it to the server.

Download address: https://codeload.github.com/macs3-project/MACS/zip/v2.1.4

Step 2 Use the SFTP tool to upload the downloaded software package to the /path/to/ MACS2 directory on the server.

4.57.6 Compiling and Installing MACS2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to decompress the MACS2 installation package:

```bash
unzip MACS-2.1.4.zip
```

Step 3 Run the following command to switch to the directory generated after decompression:

```bash
cd MACS-2.1.4
```
Step 4 Run the following commands to start the installation:

```
python setup.py build
python setup.py install
```

Step 5 Run the following commands to set environment variables and make them take effect:

```
vim /etc/profile

export PATH=/usr/bin:$PATH
export PYTHONPATH=/usr/lib64/python2.7/site-packages:$PYTHONPATH

source /etc/profile
```

----End

4.57.7 Running and Verifying MACS2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following command to check the version:

```
macs2 --version
```

Step 3 Run the following commands to run the test case:

```
cd /path/to/MACS2/MACS-2.1.4/test
sh test.sh game
```

NOTE

game is a mandatory identifier for running. For details, see test.sh.

After the commands are executed, the following information is displayed:
Step 4 Run the following commands to view logs in `game_run_50kcontigs`:

```bash
cd game_run_50kcontigs/
cat *.log
```
Step 5  Run the following commands to view logs in game_run_refinepeak:

```bash
cd game_run_refinepeak/
cat *.log
```

----End

4.57.8 More Resources

To obtain more resources, visit the MACS2 page of GitHub:

https://github.com/macs3-project/MACS

4.58 Trinity 2.11.0 Porting Guide (CentOS 7.6)

4.58.1 Introduction

Trinity, developed at the Broad Institute and the Hebrew University of Jerusalem, represents a novel method for the efficient and robust de novo reconstruction of transcriptomes from RNA-seq data. Trinity combines three independent software
modules: Inchworm, Chrysalis, and Butterfly, applied sequentially to process a large number of RNA-seq reads. Trinity divides the sequence data into many independent de Bruijn graphs, each representing the transcriptional complexity at a given gene or locus, and then processes each graph independently to extract full-length splicing isoforms and to tease apart transcripts derived from paralogous genes.

For more information about Trinity, visit the official Trinity website.

Programming language: Perl

Brief description: Gene assembly

Recommended Version

Trinity 2.11.0

4.58.2 Environment Requirements

Hardware Requirements

Table 4-286 lists the hardware requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

Software Requirements

Table 4-287 lists the software requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>boost</td>
<td>1.66.0</td>
<td><a href="https://dl.bintray.com/boostorg/release/1.66.0/source/boost_1_66_0.tar.gz">https://dl.bintray.com/boostorg/release/1.66.0/source/boost_1_66_0.tar.gz</a></td>
</tr>
<tr>
<td>salmon</td>
<td>1.4.0</td>
<td><a href="https://github.com/COMBINE-lab/salmon/archive/v1.4.0.tar.gz">https://github.com/COMBINE-lab/salmon/archive/v1.4.0.tar.gz</a></td>
</tr>
<tr>
<td>jellyfish</td>
<td>2.2.10</td>
<td><a href="https://github.com/gmarcais/Jellyfish/releases/download/v2.2.10/jellyfish-2.2.10.tar.gz">https://github.com/gmarcais/Jellyfish/releases/download/v2.2.10/jellyfish-2.2.10.tar.gz</a></td>
</tr>
<tr>
<td>htslib</td>
<td>1.11</td>
<td><a href="https://github.com/samtools/htslib/archive/1.11.tar.gz">https://github.com/samtools/htslib/archive/1.11.tar.gz</a></td>
</tr>
<tr>
<td>samtools</td>
<td>1.11</td>
<td><a href="https://github.com/samtools/samtools/archive/1.11.tar.gz">https://github.com/samtools/samtools/archive/1.11.tar.gz</a></td>
</tr>
<tr>
<td>Item</td>
<td>Version</td>
<td>Download URL</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bowtie2</td>
<td>2.3.5.1</td>
<td><a href="https://netix.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.3.5.1-source.zip">https://netix.dl.sourceforge.net/project/bowtie-bio/bowtie2/2.3.5.1-source.zip</a></td>
</tr>
<tr>
<td>sse2neno</td>
<td>master</td>
<td><a href="https://github.com/jratcliff63367/sse2neon/archive/master.zip">https://github.com/jratcliff63367/sse2neon/archive/master.zip</a></td>
</tr>
<tr>
<td>SIMDDe</td>
<td>v0.7.0-rc-1</td>
<td><a href="https://github.com/simd-everywhere/simde/archive/v0.7.0-rc-1.tar.gz">https://github.com/simd-everywhere/simde/archive/v0.7.0-rc-1.tar.gz</a></td>
</tr>
<tr>
<td>python</td>
<td>3.8</td>
<td><a href="https://www.python.org/ftp/python/3.8.0/Python-3.8.0.tgz">https://www.python.org/ftp/python/3.8.0/Python-3.8.0.tgz</a></td>
</tr>
<tr>
<td>Cython</td>
<td>0.29.21</td>
<td><a href="https://files.pythonhosted.org/packages/6c/9f/f501ba9d178aeb1f5bf7da1ad5619b207c90ac235d9859961c11829d0160/Cython-0.29.21.tar.gz">https://files.pythonhosted.org/packages/6c/9f/f501ba9d178aeb1f5bf7da1ad5619b207c90ac235d9859961c11829d0160/Cython-0.29.21.tar.gz</a></td>
</tr>
<tr>
<td>numpy</td>
<td>1.19.4</td>
<td><a href="https://files.pythonhosted.org/packages/c5/63/a48648ebc57711348420670bb074998f79828291f68aebfff1642be212ec/numpy-1.19.4.zip">https://files.pythonhosted.org/packages/c5/63/a48648ebc57711348420670bb074998f79828291f68aebfff1642be212ec/numpy-1.19.4.zip</a></td>
</tr>
<tr>
<td>Trinity</td>
<td>2.11.0</td>
<td><a href="https://github.com/trinityrnaseq/trinityrnaseq/archive/v2.11.0.tar.gz">https://github.com/trinityrnaseq/trinityrnaseq/archive/v2.11.0.tar.gz</a></td>
</tr>
<tr>
<td>Inchworm</td>
<td>master</td>
<td><a href="https://github.com/trinityrnaseq/Inchworm/archive/master.zip">https://github.com/trinityrnaseq/Inchworm/archive/master.zip</a></td>
</tr>
<tr>
<td>Chrysalis</td>
<td>master</td>
<td><a href="https://github.com/trinityrnaseq/Chrysalis/archive/8ef13ed1fa618186ec3dc04522115c99fdfacda2.zip">https://github.com/trinityrnaseq/Chrysalis/archive/8ef13ed1fa618186ec3dc04522115c99fdfacda2.zip</a></td>
</tr>
<tr>
<td>seqtk-trinity</td>
<td>master</td>
<td><a href="https://github.com/trinityrnaseq/seqtk-trinity/archive/f5e229dc164823c3ef14e8b8e2ba7d84c8d80e6c.zip">https://github.com/trinityrnaseq/seqtk-trinity/archive/f5e229dc164823c3ef14e8b8e2ba7d84c8d80e6c.zip</a></td>
</tr>
<tr>
<td>bamsifter</td>
<td>master</td>
<td><a href="https://github.com/trinityrnaseq/bamsifter/archive/f034f247d5fa1fd9847c390d99bdfd918da7d7739.zip">https://github.com/trinityrnaseq/bamsifter/archive/f034f247d5fa1fd9847c390d99bdfd918da7d7739.zip</a></td>
</tr>
<tr>
<td>Test case 1</td>
<td>SRR30508</td>
<td><a href="http://ftp.sra.ebi.ac.uk/vol1/fastq/SRR305/008/SRR3050898/SRR3050898_1.fastq.gz">http://ftp.sra.ebi.ac.uk/vol1/fastq/SRR305/008/SRR3050898/SRR3050898_1.fastq.gz</a></td>
</tr>
<tr>
<td>Test case 2</td>
<td>SRR30508</td>
<td><a href="http://ftp.sra.ebi.ac.uk/vol1/fastq/SRR305/008/SRR3050898/SRR3050898_2.fastq.gz">http://ftp.sra.ebi.ac.uk/vol1/fastq/SRR305/008/SRR3050898/SRR3050898_2.fastq.gz</a></td>
</tr>
</tbody>
</table>

**OS Requirements**

*Table 4-288* lists the OS requirements.

**Table 4-288** OS requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Item</td>
<td>Version</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.58.3 Planning the Paths for Software Porting

This section describes the software installation paths involved in the Trinity software porting.

**Table 4-289 Paths for software porting**

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning Data for Installation&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/SALMON</td>
<td>Installation path of Salmon.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>/path/to/JELLYFISH</td>
<td>Installation path of Jellyfish.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/BOWTIE2</td>
<td>Installation path of Bowtie 2.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/HTSLIB</td>
<td>Installation path of HTSlib.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>/path/to/SAMTOOLS</td>
<td>Installation path of SAMtools.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>/path/to/PYTHON</td>
<td>Installation path of Python.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>/path/to/BOOST</td>
<td>Installation path of Boost.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>/path/to/TRINITY</td>
<td>Installation path of Trinity.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>/path/to/CMAKE</td>
<td>Installation path of CMake.</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>/path/to/CASE</td>
<td>Path for storing cases.</td>
<td></td>
</tr>
</tbody>
</table>

### 4.58.4 Configuring the Compilation Environment

**Prerequisites**

The installation packages are uploaded to the server using an SFTP tool.
**Configuration Process**

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set up the basic environment.</td>
<td>For details, see &quot;Setting Up the Environment for the Single-Node Scenario&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>Install dependencies.</td>
<td>For details, see 4.58.4.1 Installing All Dependencies Using the yum Command.</td>
</tr>
<tr>
<td>3</td>
<td>Install Boost.</td>
<td>For details, see 4.58.4.2 Installing Boost.</td>
</tr>
<tr>
<td>4</td>
<td>Install CMake.</td>
<td>For details, see 4.58.4.3 Installing CMake.</td>
</tr>
<tr>
<td>5</td>
<td>Install Salmon.</td>
<td>For details, see 4.58.4.4 Installing Salmon.</td>
</tr>
<tr>
<td>6</td>
<td>Install HTSlib.</td>
<td>For details, see 4.58.4.5 Installing HTSlib.</td>
</tr>
<tr>
<td>7</td>
<td>Install SAMtools.</td>
<td>For details, see 4.58.4.6 Installing SAMtools.</td>
</tr>
<tr>
<td>8</td>
<td>Install Jellyfish.</td>
<td>For details, see 4.58.4.7 Installing Jellyfish.</td>
</tr>
<tr>
<td>9</td>
<td>Install Bowtie 2.</td>
<td>For details, see 4.58.4.8 Installing Bowtie 2.</td>
</tr>
<tr>
<td>10</td>
<td>Install Python.</td>
<td>For details, see 4.58.4.9 Installing Python.</td>
</tr>
<tr>
<td>11</td>
<td>Install Cython.</td>
<td>For details, see 4.58.4.10 Installing Cython.</td>
</tr>
<tr>
<td>12</td>
<td>Install NumPy.</td>
<td>For details, see 4.58.4.11 Installing NumPy.</td>
</tr>
</tbody>
</table>

### 4.58.4.1 Installing All Dependencies Using the yum Command

**Step 1** Use PuTTY to log in to the server as the **root** user.

**Step 2** Run the following **yum** command to install all dependencies:

```
```

----End

### 4.58.4.2 Installing Boost

**Procedure**

**Step 1** Use PuTTY to log in to the server as the **root** user.
Step 2 Use the SFTP tool to upload the Boost installation package to the `/path/to/BOOST` directory.

Step 3 Run the following command to decompress the package:

```
cd /path/to/BOOST && tar -zxf boost_1_66_0.tar.gz
```

Step 4 Run the following command to switch to the installation directory:

```
cd boost_1_66_0
```

Step 5 Run the following commands to perform compilation and installation:

```
CC=`which gcc` CXX=`which g++` ./bootstrap.sh --prefix=/path/to/BOOST
./b2 -j 32
./b2 install
-----End
```

### 4.58.4.3 Installing CMake

**Procedure**

Step 1 Use PuTTY to log in to the server as the `root` user.

Step 2 Use the SFTP tool to upload the CMake installation package to the `/path/to/CMAKE` directory.

Step 3 Run the following command to decompress the package:

```
cd /path/to/CMAKE && tar -zxf cmake-3.17.3.tar.gz
```

Step 4 Run the following command to switch to the installation directory:

```
cd cmake-3.17.3
```

Step 5 Run the following commands to perform compilation and installation:

```
CC=`which gcc` CXX=`which g++` ./bootstrap --prefix=/path/to/CMAKE
gmake && gmake install
-----End
```

### 4.58.4.4 Installing Salmon

**Procedure**

Step 1 Use PuTTY to log in to the server as the `root` user.

Step 2 Use the SFTP tool to upload the Salmon installation package to the `/path/to/SALMON` directory.

Step 3 Run the following command to set the environment variable:

```
export PATH=/path/to/CMAKE/bin:$PATH
```

Step 4 Run the following command to decompress the package:
cd /path/to/SALMON && tar -zxf v1.4.0.tar.gz

**Step 5** Run the following command to switch to the installation directory:

cd salmon-1.4.0

**Step 6** Run the following command to create and enter the build directory:

mkdir build && cd build

**Step 7** Run the following commands to perform compilation and installation:

```
CC=`which gcc` CXX=`which g++` cmake -DNO_IPO=TRUE -DCMAKE_C_FLAGS="-O3 -march=armv8.2-a -mtune=tsv110" -DCMAKE_CXX_FLAGS="-O3 -march=armv8.2-a -mtune=tsv110" -DCMAKE_INSTALL_PREFIX=/path/to/SALMON -DBOOST_INCLUDEDIR=/path/to/BOOST/include -DBOOST_LIBRARYDIR=/path/to/BOOST/lib ..
make -j && make install
```

----End

### 4.58.4.5 Installing HTSlib

#### Procedure

**Step 1** Use PuTTY to log in to the server as the root user.

**Step 2** Use the SFTP tool to upload the HTSlib installation package to the /path/to/htslib directory.

**Step 3** Run the following command to decompress the package:

```
cd /path/to/HTSLIB && tar -zxf htslib-1.11.tar.gz
```

**Step 4** Run the following command to rename and go to the installation directory:

```
mv htslib-1.11 htslib && cd htslib
```

**Step 5** Run the following commands to perform configuration:

```
autoheader
autoconf
```

```
import re
fl = open("./configure", "r")
for l in fl.readlines():
    l = l.replace('-', '-O3 -march=armv8.2-a -mtune=tsv110')
    l = l.replace('-O2', '-O3 -march=armv8.2-a -mtune=tsv110')
    print(l)
fl.close()
```

```
CC=`which gcc` ./configure --prefix=/path/to/HTSLIB
```

**Step 6** Run the following commands to perform compilation and installation:

```
make -j 16
make install
```

----End
4.58.4.6 Installing SAMtools

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Use the SFTP tool to upload the SAMtools installation package to the /path/to/SAMTOOLS directory.

Step 3  Run the following command to decompress the package:
        `cd /path/to/SAMTOOLS & & tar -zxf samtools-1.11.tar.gz`

Step 4  Run the following command to switch to the installation directory:
        `cd samtools-1.11`

Step 5  Run the following commands to perform configuration:
        `autoheader`
        `autoconf`
        `sed -i 's/-O2/-O3 -march=armv8.2-a -mtune=tsv110/g' `grep -lr "-O2" ./``
        CC=`which gcc` `./configure --prefix=/path/to/SAMTOOLS --with-htslib=/path/to/HTSLIB`

Step 6  Run the following commands to perform compilation and installation:
        `make -j 16`
        `make install`

----End

4.58.4.7 Installing Jellyfish

Procedure

Step 1  Use PuTTY to log in to the server as the root user.

Step 2  Use the SFTP tool to upload the Jellyfish installation package to the /path/to/JELLYFISH directory.

Step 3  Run the following command to decompress the package:
        `cd /path/to/JELLYFISH & & tar -zxf jellyfish-2.2.10.tar.gz`

Step 4  Run the following command to switch to the directory generated after decompression:
        `cd jellyfish-2.2.10`

Step 5  Run the following command to perform configuration:
        `CC=`which gcc` `CXX=`which g++` `./configure --prefix=/path/to/JELLYFISH` 
        `CXXFLAGS='-O3 -std=c++11 -march=armv8.2-a -mtune=tsv110 -flto' CFLAGS=''-O3 -march=armv8.2-a -mtune=tsv110 -flto'`
Step 6 Run the following command to perform installation and compilation.

```
make -j && make install
```

4.58.4.8 Installing Bowtie 2

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use the SFTP tool to upload the installation packages of Bowtie 2, SIMDe, and sse2neon to the /path/to/BOWTIE2 directory.

Step 3 Run the following commands to decompress the installation package and go to the directory generated after the decompression:

```
cd /path/to/BOWTIE2
unzip bowtie2-2.3.5.1-source.zip
tar -zxf v0.7.0-rc-1.tar.gz
unzip master.zip
cd bowtie2-2.3.5.1
```

Step 4 Run the following command to copy the SIMDe files to Bowtie 2:

```
cp -a ../simde-0.7.0-rc-1/simde third_party/simde
```

Step 5 Run the following command to copy the sse2neon.h file to Bowtie 2:

```
cp -a ../sse2neon-master/SSE2NEON.h third_party/simde/simde/x86/
```

Step 6 Run the following commands to perform compilation:

```
CC=`which gcc` CXX=`which g++` CXXFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' CFLAGS='-O3 -march=armv8.2-a -mtune=tsv110' make NO_TBB=1 POPCNT_CAPABILITY=0 all
```

4.58.4.9 Installing Python

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use the SFTP tool to upload the Python installation package to the /path/to/PYTHON directory.

Step 3 Run the following commands to switch to the installation directory and decompress the installation package:

```
cd /path/to/PYTHON

tar -zxf Python-3.8.0.tgz && cd Python-3.8.0
```
Step 4 Run the following command to configure compilation information:

```
./configure --prefix=/path/to/PYTHON --enable-shared --enable-loadable-sqlite-extensions --enable-optimization
```

Step 5 Run the following command to perform compilation and installation:

```
make -j && make install
```

---End

### 4.58.4.10 Installing Cython

**Procedure**

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use the SFTP tool to upload the Cython installation package to the `PATH/to/PYTHON` directory.

Step 3 Run the following commands to set environment variables:

```
export PATH=/path/to/PYTHON/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON/lib:$LD_LIBRARY_PATH
```

Step 4 Run the following command to decompress the installation package and go to the installation directory:

```
tar -zxvf Cython-0.29.21.tar.gz && cd Cython-0.29.21
```

Step 5 Run the following command to perform compilation and installation:

```
python3 setup.py build && python3 setup.py install
```

---End

### 4.58.4.11 Installing NumPy

**Procedure**

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Use the SFTP tool to upload the NumPy installation package to the `PATH/to/PYTHON` directory.

Step 3 Run the following commands to set environment variables:

```
export PATH=/path/to/PYTHON/bin:$PATH
export LD_LIBRARY_PATH=/path/to/PYTHON/lib:$LD_LIBRARY_PATH
```

Step 4 Run the following command to decompress the installation package and go to the installation directory:

```
unzip numpy-1.19.4.zip && cd numpy-1.19.4
```

Step 5 Run the following command to perform compilation and installation:
4.58.5 Obtaining the Source Code

Procedure

Step 1 Download the installation packages of Trinity, Inchworm, Chrysalis, seqtk-trinity, bamsifter and HTSlib.

Download Trinity at [https://github.com/trinityrnaseq/trinityrnaseq/archive/v2.11.0.tar.gz](https://github.com/trinityrnaseq/trinityrnaseq/archive/v2.11.0.tar.gz)

Download Inchworm at [https://github.com/trinityrnaseq/Inchworm/archive/master.zip](https://github.com/trinityrnaseq/Inchworm/archive/master.zip)

Download Chrysalis at [https://github.com/trinityrnaseq/Chrysalis/archive/8ef13ed1fa618186ec3dc04522115c99f9facda2.zip](https://github.com/trinityrnaseq/Chrysalis/archive/8ef13ed1fa618186ec3dc04522115c99f9facda2.zip)

Download seqtk-trinity at [https://github.com/trinityrnaseq/seqtk-trinity/archive/f5e229dc164823c3ef14e8b8e2ba7d84c8d80e6c.zip](https://github.com/trinityrnaseq/seqtk-trinity/archive/f5e229dc164823c3ef14e8b8e2ba7d84c8d80e6c.zip)

Download bamsifter at [https://github.com/trinityrnaseq/bamsifter/archive/f034f247d5fa1fd9847c390d99b9d918da7d7739.zip](https://github.com/trinityrnaseq/bamsifter/archive/f034f247d5fa1fd9847c390d99b9d918da7d7739.zip)

Download HTSlib at [https://github.com/samtools/htslib/archive/1.11.zip](https://github.com/samtools/htslib/archive/1.11.zip)

Step 2 Use the SFTP tool to upload the installation packages of Trinity, Inchworm, and Chrysalis to the /path/to/TRINITY directory.

4.58.6 Compiling and Installing Trinity

Procedure

Step 1 Run the following commands to go to the installation directory and decompress the software package:

```
cd /path/to/TRINITY
tar -zxf v2.11.0.tar.gz
```

```
unzip master.zip
```

```
unzip 8ef13ed1fa618186ec3dc04522115c99f9facda2.zip
```

```
unzip f034f247d5fa1fd9847c390d99b9d918da7d7739.zip
```

```
unzip f5e229dc164823c3ef14e8b8e2ba7d84c8d80e6c.zip
```

```
unzip 1.11.zip
```

Step 2 Run the following commands to save the source codes of Inchworm, Chrysalis, seqtk-trinity, and bamsifter to the corresponding directories in the Trinity source code:
Step 3 Run the following command to decompress the ParaFly-0.1.0.tar.gz package:
```
cd trinityrnaseq-2.11.0/trinity-plugins && tar -zxf ParaFly-0.1.0.tar.gz
```

Step 4 Run the following commands to replace the parameter `-m64` by `-mabi=lp64`:
```
cd /path/to/TRINITY/trinityrnaseq-2.11.0
sed -i '/tar -zxvf \${PARAFLY_CODE}.tar.gz/d' trinity-plugins/Makefile
sed -i 's/\-m64/\-mabi=lp64/g' `grep -rl "\-m64" ./`
```

Step 5 Run the following command to add compilation optimization parameters:
```
sed -i 's/\$make/cmake -DCMAKE_CXX_FLAGS="-O3 -march=armv8.2-a -mtune=tsv110"/g' ./Chrysalis/Makefile .Inchworm/Makefile
```
```
sed -i 's/\-fopenmp/-O3 -march=armv8.2-a -mtune=tsv110 -fopenmp/g' ./trinity-plugins/Makefile
```

Step 6 Run the following command to set the environment variable:
```
export PATH=/path/to/CMAKE/bin:$PATH
```

Step 7 Run the following command to perform compilation and installation:
```
make -j
```

If the information shown in the following figure is displayed, the installation is successful.

```
Performing Unit Tests of Build

Inchworm: has been Installed Properly
Chrysalis: has been Installed Properly
QuantifyGraph: has been Installed Properly
GraphFromFasta: has been Installed Properly
ReadsToTranscripts: has been Installed Properly
parafly: has been Installed Properly

-----End
```
4.58.7 Running and Verifying Trinity

Procedure

Step 1 Use PuTTY to log in to the server as the root user.

Step 2 Run the following commands to set environment variables:

- `export PATH=/path/to/SAMTOOLS/bin:$PATH`
- `export PATH=/path/to/SALMON/bin:$PATH`
- `export PATH=/path/to/JELLYFISH/bin:$PATH`
- `export PATH=/path/to/BOWTIE2/bowtie2-2.3.5.1:$PATH`
- `export LD_LIBRARY_PATH=/path/to/HTSLIB/lib:$LD_LIBRARY_PATH`
- `export PATH=/path/to/PYTHON/bin:$PATH`
- `export PATH=/path/to/TRINITY/trinityrnaseq-2.11.0:$PATH`
- `export LD_LIBRARY_PATH=/path/to/HTSLIB/lib:$LD_LIBRARY_PATH`

Step 3 Run the following command to switch to the case directory and decompress the case:

- `cd /path/to/CASE`
- `gzip -d SRR3050898_1.fastq.gz && gzip -d SRR3050898_2.fastq.gz`

Step 4 Run the following command to change the maximum size of the Java heap to 30 GB:

- `sed -i 's/bflyHeapSpaceMax = "10G"/bflyHeapSpaceMax = "30G"/g' /path/to/TRINITY/trinityrnaseq-2.11.0/Trinity`

Step 5 Run the following command to run the case:

- `{ time Trinity --seqType fq --left SRR3050898_1.fastq --right SRR3050898_2.fastq --CPU 128 --max_memory 200G --output trinity_kp_`date '+%Y%m%d%H%M%S'`; } 2>&1 & tee -a `hostname`_`date '+%Y%m%d%H%M%S'`.log`

**Figure 4-30** shows an example of the running result.
4.59 LDBlockshow 1.36 Porting Guide (CentOS 7.6)

4.59.1 Introduction

LDBlockShow is a fast and effective tool to generate linkage disequilibrium (LD) heatmap from VCF files. It is more time and memory saving than other current tools. LDBlockShow can generate the plots of LD heatmap and interested statistics or annotation results simultaneously. In addition, it also supports subgroup analysis.

For more information about LDBlockShow, visit the official LDBlockShow website.

Language: Perl/HTML/C++/shell

One-sentence description: a quick and effective tool for generating LD heat maps from VCF files

Open-source protocol: MIT License

Recommended Version

The recommended version is LDBlockshow1.36.

4.59.2 Environment Requirements

Hardware Requirements

Table 4-291 lists the hardware requirements.
### Table 4-291 Hardware requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Kunpeng 920</td>
</tr>
</tbody>
</table>

### Software Requirements

Table 4-292 lists the software requirements.

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
<th>Installation Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDBlockShow</td>
<td>1.36</td>
<td><a href="https://github.com/BGI-shenzhen/LDBlockShow/archive/v1.36.tar.gz">https://github.com/BGI-shenzhen/LDBlockShow/archive/v1.36.tar.gz</a></td>
</tr>
<tr>
<td>plink</td>
<td>1.9</td>
<td><a href="https://github.com/chrchang/plink-ng/archive/master.tar.gz">https://github.com/chrchang/plink-ng/archive/master.tar.gz</a></td>
</tr>
<tr>
<td>Test case</td>
<td>example</td>
<td>Built-in Example1/Example2/Example3/Example4</td>
</tr>
</tbody>
</table>

### OS Requirements

Table 4-293 lists the OS requirements.

<table>
<thead>
<tr>
<th>Item</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7.6</td>
<td><a href="https://www.centos.org/download/">https://www.centos.org/download/</a></td>
</tr>
<tr>
<td>Kernel</td>
<td>4.14.0-115</td>
<td>Included in the OS image.</td>
</tr>
</tbody>
</table>

### 4.59.3 Planning Paths for Software Porting

This section describes the software installation paths involved in the LDBlockshow software porting.
Table 4-294 Paths for software porting

<table>
<thead>
<tr>
<th>No.</th>
<th>Software Installation Path</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Installation path of each software package required for setting up the basic environment.</td>
<td>For details, see &quot;Planning the Installation Paths&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
<tr>
<td>2</td>
<td>/path/to/LDBLOCKSHOW</td>
<td>Installation path of LDBlockshow.</td>
<td>The installation paths listed in this table are only for reference. You are advised to deploy the software in a shared path. All the paths used in commands in this document need to be replaced with the actual paths planned during the installation process.</td>
</tr>
<tr>
<td>3</td>
<td>/path/to/SVG</td>
<td>Installation path of SVG.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>/path/to/PLINK</td>
<td>Installation path of PLINK.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/path/to/ZLIB</td>
<td>Installation path of zlib.</td>
<td></td>
</tr>
</tbody>
</table>

4.59.4 Obtaining the Source Code

Procedure

**Step 1** Run the following command to download the SVG installation package:

```
wget https://www.cpan.org/modules/by-module/SVG/SVG-2.85.tar.gz
```

**Step 2** Run the following command to download the LDBlockShow installation package:

```
wget https://github.com/BGI-shenzhen/LDBlockShow/archive/v1.36.tar.gz --no-check-certificate
```

**Step 3** Run the following command to download the PLINK installation package:

```
wget https://github.com/chrchang/plink-ng/archive/master.tar.gz --no-check-certificate
```

**Step 4** Run the following command to download the zlib installation package:

```
wget http://zlib.net/zlib-1.2.11.tar.gz
```

----End

4.59.5 Configuring the Compilation Environment

Prerequisites

Installation packages are uploaded to a server using an SFTP tool.
Configuration Process

<table>
<thead>
<tr>
<th>No.</th>
<th>Configuration Item</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Basic environment setup</td>
<td>For details, see &quot;Setting Up the Single-Node System Environment&quot; in the HPC Solution Basic Environment Setup Guide.</td>
</tr>
</tbody>
</table>

4.59.6 Compiling and Installing LDBlockShow

Procedure

Step 1  Run the following command to install the basic dependency package:

```
yum install zlib perl-ExtUtils-CBuilder perl-ExtUtils-MakeMaker
```

Step 2  Run the following commands to specify a compiler:

```
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`
```

Step 3  Run the following commands to install SVG:

```
tar -xzvf SVG-2.85.tar.gz

cd SVG-2.85
perl Makefile.PL
make
make install
```

Step 4  Run the following commands to install LDBlockShow:

```
tar -zxvf v1.36.tar.gz

cd LDBlockShow-1.36
chmod +x configure
./configure
make
```

Step 5  Run the following commands to install PLINK:

```
yum install blas-devel lapack-devel atlas-devel perl-Digest-SHA -y
ln -s /usr/lib64/atlas/libatlas.so.3.10 /usr/lib64/atlas/libatlas.so
```
In -s /usr/lib64/atlas/libsatlas.so.3.10 /usr/lib64/atlas/libcblas.so

tar -zxf master.tar.gz

mv zlib-1.2.11.tar.gz plink-ng-master

cd plink-ng-master/1.9 && sed -i '0,/curl/s/curl/# curl/g' plink_first_compile

./plink_first_compile

cp plink /path/to/LDBLOCKSHOW/LDBlockShow-1.36/bin/

cp: overwrite '/path/to/LDBLOCKSHOW/LDBlockShow-1.36/bin/plink'? y

-----End

4.59.7 Running and Verifying LDBlockShow

Procedure

Step 1  Run the following commands to start LDBlockShow:

    cd /path/to/LDBLOCKSHOW/LDBlockShow-1.36/example/Example1
    sh run.sh

The output is displayed as follows:

    Start Time : Thu Dec 24 17:09:54 CST 2020
    #Detected VCF File is phased file with '1', Read VCF in Phase mode
    #Start Region Cal... : chr11 24100000 24200000; In This Region TotalSNP Number is 7
    Find blocks...
    Start draw... SVG info: SNPNumber :7 , SVG (width,height) = (455,385)
    Convert SVG ---> PNG ...
    ALL done
    End Time : Thu Dec 24 17:09:54 CST 2020

-----End
## Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2021-07-23 | This issue is the fourteenth official release.  
- Modified the commands in step 2 of section "Running and Verifying NEMO" in the *NEMO V3.6 Porting Guide (CentOS 7.6)*.  
- Deleted the `-lnetcdff` parameter from step 4 of section "Installing SMOKE" in the *SMOKE 4.7 Porting Guide (CentOS 7.6)*.  
- Modified the description in step 6 of section "Running and Verifying OpenFOAM" in the *OpenFOAM 1906 Porting Guide (CentOS 7.6)*. |
| 2021-06-25 | This issue is the thirteenth official release.  
Added 10 porting guides:  
- Manufacturing  
  - Code_Aster 14.6.0 Porting Guide (CentOS 7.6)  
- Government HPC  
  - CPMD 4.1 Porting Guide (CentOS 7.6)  
  - ESPResSo 4.1.4 Porting Guide (CentOS 7.6)  
  - MOOSE Framework 1.0.0 Porting Guide (CentOS 7.6)  
  - MPB 1.11.1 and Meep 1.17.1 Porting Guide (CentOS 7.6)  
  - Octopus 10.3 Porting Guide (CentOS 7.6)  
  - Psi4 1.3.2 Porting Guide (CentOS 7.6)  
  - SIESTA 4.0.2 Porting Guide (CentOS 7.6)  
  - Wannier90 3.1.0 Porting Guide (CentOS 7.6)  
  - WanT 2.6.1 Porting Guide (CentOS 7.6) |
## Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2021-06-09    | This issue is the twelfth official release.  
● Modified step 6 of section "Running and Verifying OpenFOAM" in the *OpenFOAM 1906 Porting Guide (CentOS 7.6).*  
● Modified step 3 of section "Installing Bowtie 2" in the *Bowtie2 2.4.1 Porting Guide (CentOS 7.6).*  
● Modified step 4 of section "Compiling and Installing pblast" in the *pblast 2.1 Porting Guide (CentOS 7.6).* |
| 2021-06-02    | This issue is the eleventh official release.  
● Modified "Environment Requirements" in the *WRF 3.8.1 Porting Guide (CentOS 7.6).*  
● Modified section "Installing Libint" in the *CP2K 7.1 Porting Guide (CentOS 7.6).* |
| 2021-05-28    | This issue is the tenth official release.  
Recategorized the "DNA sequencing" industry component into the "Other" sector. |
| 2021-05-18    | This issue is the ninth official release.  
Changed "Supercomputing" to "Government HPC." |
| 2021-04-29    | This issue is the eighth official release,  
● Modified sections "Environment Requirements", "Installing Eigen", and "Compiling and Installing Cufflinks" in the *Cufflinks 2.2.1 Porting Guide (CentOS 7.6).*  
● Modified sections "Paths for Software Porting", "Installing Maker", and "Running and Verifying SNAP" in the *SNAP 2013-11-29 Porting Guide (CentOS 7.6).*  
● Modified sections "Environment Requirements", "Installing HDF5", and "Compiling and Installing FVCOM" in the *FVCOM 4.1 Porting Guide (CentOS 7.6).* |
| 2021-04-13    | This issue is the seventh official release.  
Changed the solution name from "Kunpeng HPC Solution" to "Kunpeng BoostKit for HPC" in the *Progressive Knowledge.* |
| 2020-12-20    | This issue is the sixth official release.  
Added the porting guides of 23 components for DNA sequencing, 6 components for meteorology, and 1 component for supercomputing. |
| 2020-09-20    | This issue is the fifth official release.  
Added the porting guides of 3 components for education and scientific research, 5 components for DNA sequencing, 2 components for meteorology, and 1 component for manufacturing. |
<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020-08-20</td>
<td>This issue is the fourth official release. Updated the <em>WRF 3.8.1 Porting Guide (CentOS 7.6)</em> for meteorology, <em>Cufflinks 2.2.1 Porting Guide (CentOS 7.6)</em> for DNA sequencing, and <em>MiniFE 2.2.0 Porting Guide (CentOS 7.6)</em> for other industries.</td>
</tr>
<tr>
<td>2020-07-20</td>
<td>This issue is the third official release. Added the porting guides of 14 components for DNA sequencing and 1 component for manufacturing.</td>
</tr>
<tr>
<td>2020-04-20</td>
<td>This issue is the second official release. Added the porting guides of 14 components for industries such as education and scientific research, DNA sequencing, and meteorology.</td>
</tr>
<tr>
<td>2020-03-20</td>
<td>This issue is the first official release.</td>
</tr>
</tbody>
</table>