HPC 解决方案

安装指南

文档版本	01
发布日期	2020-03-20

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 HPC 解决方案 基础环境搭建指南

1.1 简介

1.1.1 使用介绍

此文档用于HPC解决方案行业应用在鲲鹏计算平台上进行软件移植前的基础环境准备，下文的操作都在服务器的OS已完成安装前提下进行。

1.1.2 场景介绍

此文档主要使用于HPC两种场景，分别是：

1. 集群场景（IB网络），HPC绝大部分的应用都可以支持MPI并行化，IB网络是目前HPC场景的主流网络方案；
2. 单机场景，应用不支持多节点并行化，例如基因行业中的大部分应用。

1.2 环境要求

硬件要求

硬件要求如**表1-1**所示。

表 1-1 硬件要求

项目	说明
CPU	Kunpeng 920

操作系统要求

操作系统要求如表1-2所示。

表 1-2 操作系统要求

项目	版本	下载地址
CentOS	7.6	https://www.centos.org/download/
Kernel	4.14.0-115	包含在操作系统镜像中。

其他软件要求

其他软件要求如表1-3所示。

表 1-3 其他软件要求

项目	安装包	下载地址
GNU	gcc-9.3.0.tar.xz	https://ftp.gnu.org/gnu/gcc/gcc-9.3.0/
GMP	gmp-6.1.0.tar.bz2	http://gcc.gnu.org/pub/gcc/infrastructure/
MPC	mpc-1.0.3.tar.gz	http://gcc.gnu.org/pub/gcc/infrastructure/
MPFR	mpfr-3.1.4.tar.bz2	http://gcc.gnu.org/pub/gcc/infrastructure/
OpenMPI	openmpi-4.0.3.tar.gz	https://download.open-mpi.org/release/open-mpi/v4.0/openmpi-4.0.3.tar.gz
IB驱动	MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6alternate-aarch64.tgz	https://www.mellanox.com/page/products_dyn?product_family=26&mtag=linux_sw_drivers

1.3 安装规划数据

本章节给出HPC解决方案基础环境搭建过程中涉及的相关软件安装规划路径的用途及详细说明。

表 1-4 移植规划数据

序号	软件安装规划路径	用途	说明
1	*/path/to/GMP*	GMP 6.1.0的安装规划路径。	这里的安装规划路径只是一个举例说明，建议部署在共享路径中。现网需要根据实际情况调整，后续章节凡是遇到安装路径的命令，都以现网实际规划的安装路径为准进行替换，不再单独说明。
2	*/path/to/MPC*	MPC 1.0.3的安装规划路径。	
3	*/path/to/MPFR*	MPFR 3.1.4的安装规划路径。	
4	*/path/to/GNU*	GNU 9.3.0的安装规划路径。	
5	*/path/to/OPENMPI*	OpenMPI 4.0.1的安装规划路径。	
6	*/path/to/INFINIBAND*	IB网卡驱动的安装包存放路径。	
7	*/path/to/ISO*	OS镜像包的存放路径。	

1.4 集群场景环境搭建

本章节给出在集群场景（IB网络）下，搭建基础环境的步骤。

前提条件

使用sftp工具将所需安装包上传至服务器对应的安装目录。

1.4.1 本地 yum 源配置

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令新建“/cdrom”目录。

mkdir /cdrom

步骤3 执行以下命令将操作系统的镜像包挂载到“/cdrom”目录。

mount */path/to/ISO*/**CentOS-7-aarch64-Everything-1810.iso /cdrom**

步骤4 执行以下命令清除“/etc/yum.repos.d”目录下的配置文件。

cd /etc/yum.repos.d

mkdir bak

mv *.repo bak

步骤5 执行以下命令新建yum源配置文件。

1. **vi /etc/yum.repos.d/CentOS-base.repo**
2. 按“**i**”进入编辑模式，添加如下内容。

```
[CentOS7.6-source]
Name=CentOS 7.6 Repo
baseurl=file:///cdrom
enabled=1
gpgcheck=0
```

3. 按“**Esc**”键，输入**:wq!**，按“**Enter**”保存并退出编辑。

----结束

1.4.2 GMP 安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令解压GMP安装包。

tar -xvf gmp-6.1.0.tar.bz2

步骤3 执行以下命令进入GMP源码目录。

cd gmp-6.1.0

步骤4 执行以下命令进行编译安装。

./configure --prefix=*/path/to/GMP*

make

make install

步骤5 执行以下命令加载环境变量。

export LD_LIBRARY_PATH=*/path/to/GMP***/lib:$LD_LIBRARY_PATH**

----结束

1.4.3 MPFR 安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令解压MPFR安装包。

tar -xvf mpfr-3.1.4.tar.bz2

步骤3 执行以下命令进入MPFR源码目录。

cd mpfr-3.1.4

步骤4 执行以下命令进行编译安装。

./configure --prefix=*/path/to/MPFR* **--with-gmp=***/path/to/GMP*

make

make install

步骤5 执行以下命令加载环境变量。

export LD_LIBRARY_PATH=*/path/to/MPFR***/lib:$LD_LIBRARY_PATH**

----结束

1.4.4 MPC 安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令解压MPC安装包。

tar -xvf mpc-1.0.3.tar.gz

步骤3 执行以下命令进入MPC源码目录。

cd mpc-1.0.3

步骤4 执行以下命令进行编译安装。

./configure --prefix=*/path/to/MPC* **--with-gmp=***/path/to/GMP* **--with-mpfr=***/path/to/MPFR*

make

make install

步骤5 执行以下命令加载环境变量。

export LD_LIBRARY_PATH=*/path/to/MPC***/lib:$LD_LIBRARY_PATH**

----结束

1.4.5 GNU 安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令解压GNU安装包。

tar -vxf gcc-9.3.0.tar.xz

步骤3 执行以下命令进入GNU源码目录。

cd gcc-9.3.0

步骤4 执行以下命令进行编译安装。

./configure --disable-multilib --enable-languages="c,c++,fortran" --prefix=*/path/to/GNU* **--disable-static --enable-shared --with-gmp=** */path/to/GMP* **--with-mpfr=***/path/to/MPFR* **--with-mpc=***/path/to/MPC*

make

make install

步骤5 执行以下命令加载环境变量。

export PATH=*/path/to/GNU*/bin:$ PATH

export LD_LIBRARY_PATH=*/path/to/GNU*/lib:$LD_LIBRARY_PATH

----结束

1.4.6 IB 网卡驱动安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令安装系统依赖包。

yum install tcsh tcl lsof tk -y

步骤3 执行以下命令重新编译IB驱动。

tar -xzvf MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6alternate-aarch64.tgz

RPM_BUILD_NCPUS=16 ./mlnxofedinstall --add-kernel-support-build-only --without-depcheck --skip-distro-check

安装完成后，会在“/tmp”目录下生成新的安装包“MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6alternate-ext.tgz”。

步骤4 执行以下命令解压安装包。

cd /tmp/MLNX_OFED_LINUX-4.6-1.0.1.1-4.14.0-115.el7a.0.1.aarch64

tar -xvf MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6alternate-ext.tgz

步骤5 执行以下命令安装Infiniband驱动。

cd MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6alternate-ext

./mlnxofedinstall

步骤6 执行以下命令重启服务器。

reboot

----结束

1.4.7 OpenMPI 安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令安装系统依赖包。

yum install libxml2* systemd-devel.aarch64 numa* -y

步骤3 执行以下命令解压OpenMPI安装包。

tar -zxvf openmpi-4.0.3.tar.gz

步骤4 执行以下命令进入源码目录。

cd openmpi-4.0.3

步骤5 执行以下命令编译安装OpenMPI。

./configure --prefix=*/path/to/OPENMPI* **--enable-pretty-print-stacktrace --enable-orterun-prefix-by-default --with-knem=/opt/knem-1.1.3.90mlnx1/ --with-hcoll=/opt/mellanox/hcoll/ --with-cma --with-ucx --enable-mpi1-compatibility**

步骤6 执行以下命令加载环境变量。

export PATH=*/path/to/OPENMPI***/bin:$PATH**

export LD_LIBRARY_PATH=*/path/to/OPENMPI***/lib:$LD_LIBRARY_PATH**

----结束

1.5 单机场景环境搭建

本章节给出在单机场景下，搭建基础环境的步骤。

前提条件

使用sftp工具将所需安装包上传至服务器对应的安装目录。

1.5.1 本地 yum 源配置

参考**1.4.1 本地yum源配置**。

1.5.2 GMP 安装

参考**1.4.2 GMP安装**。

1.5.3 MPFR 安装

参考**1.4.3 MPFR安装**。

1.5.4 MPC 安装

参考**1.4.4 MPC安装**。

1.5.5 GNU 安装

参考**1.4.5 GNU安装**。

2 GNU 9.1 安装指南

2.1 介绍

简要介绍

GNU是一个开源的开发工具链，其中包括GCC编译器、汇编器、链接器等开源工具部件。GNU编译器从4.9版本开始支持TaiShan服务器，并且在GNU9版本后加入了核心优化相关功能，能友好且高效性的运用在TaiShan服务器上。

建议的版本

建议使用版本为“GNU 9.1”。

2.2 环境信息

硬件要求

硬件要求如**表2-1**所示。

表 2-1 硬件要求

项目	说明
CPU	Kunpeng 920

软件要求

软件要求如表2-2所示。

表 2-2 软件要求

项目	版本	下载地址
GMP	6.1.0	http://gcc.gnu.org/pub/gcc/infrastructure/
MPFR	3.1.4	
MPC	1.0.3	
GNU	9.1.0	https://ftp.gnu.org/gnu/gcc/gcc-9.1.0/
OpenMPI	4.0.1	https://www.open-mpi.org/software/ompi/v4.0/

操作系统要求

操作系统要求如表2-3所示。

表 2-3 操作系统要求

项目	版本	下载地址
CentOS	7.6	https://www.centos.org/download/

2.3 部署规划数据

本章节给出GNU软件在安装部署过程中涉及到的相关软件安装规划路径的用途及详细说明。

表 2-4 部署规划数据

序号	软件安装规划路径	用途	说明
1	*/path/to/GMP*	GMP的安装规划路径。	这里的安装规划路径只是一个举例说明，建议部署在共享路径中。现网需要根据实际情况调整，后续章节凡是遇到安装路径的命令，都以现网实际规划的安装路径为准进行替换，不再单独说明。
2	*/path/to/MPFR*	MPFR的安装规划路径。	
3	*/path/to/MPC*	MPC的安装规划路径。	

序号	软件安装规划路径	用途	说明
4	*/path/to/GNU*	GNU的安装规划路径。	
5	*/path/to/OPENMPI*	OPENMPI的安装规划路径。	

2.4 部署 GNU9.1

2.4.1 下载安装包

操作步骤

步骤1 下载以下安装包：

- GMP安装包：“gmp-6.1.0.tar.bz2”
- MPFR安装包：“mpfr-3.1.4.tar.bz2”
- MPC安装包：“mpc-1.0.3.tar.gz”

下载地址：**http://gcc.gnu.org/pub/gcc/infrastructure/**。

步骤2 下载GNU安装包“gcc-9.1.0.tar.xz”。

下载地址：**https://ftp.gnu.org/gnu/gcc/gcc-9.1.0/**

步骤3 下载OpenMPI安装包“openmpi-4.0.1.tar.gz”。

下载地址：**https://www.open-mpi.org/software/ompi/v4.0/**

步骤4 使用SFTP工具：

- 将GMP安装包上传至服务器“*/path/to/GMP*”目录。
- 将MPFR安装包上传至服务器“*/path/to/MPFR*”目录。
- 将MPC安装包上传至服务器“*/path/to/MPC*”目录。
- 将GNU安装包上传至服务器“*/path/to/GNU*”目录。
- 将OpenMPI安装包上传至服务器“*/path/to/OPENMPI*”目录。

----结束

2.4.2 GMP 编译安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令解压安装包。

tar -vxf gmp-6.1.0.tar.bz2

步骤3 执行以下命令进入解压文件路径。

cd gmp-6.1.0/

步骤4 执行以下命令进行编译安装。

./configure --prefix=*/path/to/GMP*

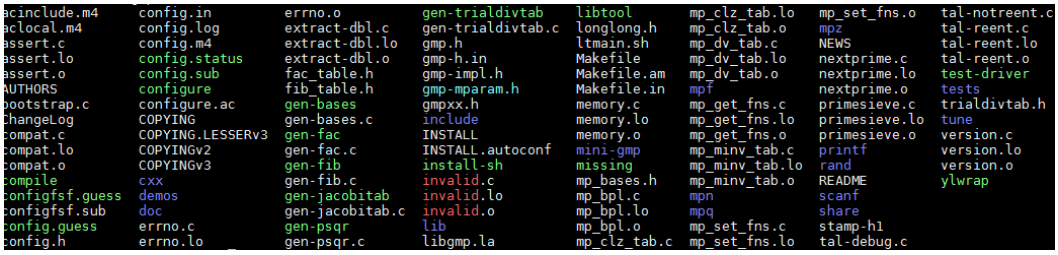
make

make install

步骤5 执行以下命令加载环境变量。

export LD_LIBRARY_PATH=*/path/to/GMP***/gmp-6.1.0/lib:$LD_LIBRARY_PATH**

编译安装成功如下图所示。



----结束

2.4.3 MPFR 编译安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令解压安装包。

tar -xvf mpfr-3.1.4.tar.bz2

步骤3 执行以下命令进入解压文件路径。

cd mpfr-3.1.4/

步骤4 执行以下命令进行编译安装。

./configure --prefix=*/path/to/MPFR***/mpfr-3.1.4 --with-gmp=***/path/to/GMP***/gmp-6.1.0**

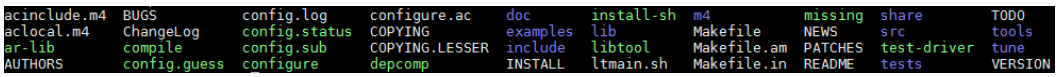
make

make install

步骤5 执行以下命令加载环境变量。

export LD_LIBRARY_PATH=*/path/to/MPFR***/mpfr-3.1.4/lib:$LD_LIBRARY_PATH**

编译安装成功如下图所示。



----结束

2.4.4 MPC 编译安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令解压安装包。

tar -zvxf mpc-1.0.3.tar.gz

步骤3 执行以下命令进入解压文件路径。

cd mpc-1.0.3/

步骤4 执行以下命令进行编译安装。

./configure --prefix=*/path/to/MPC*/**mpc-1.0.3 --with-gmp=***/path/to/GMP*/**gmp-6.1.0 --with-mpfr=***/path/to/MPFR*/**mpfr-3.1.4**

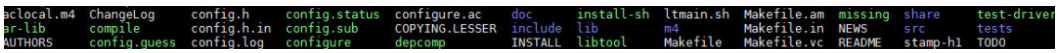
make

make install

步骤5 执行以下命令加载环境变量。

export LD_LIBRARY_PATH=*/path/to/MPC*/**mpc-1.0.3/lib:$LD_LIBRARY_PATH**

编译安装成功如下图所示。



----结束

2.4.5 GNU9.1 编译安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令解压安装包。

tar -vxf gcc-9.1.0.tar.xz

步骤3 执行以下命令进入解压文件路径。

cd gcc-9.1.0/

步骤4 执行以下命令创建obj文件。

mkdir obj

步骤5 执行以下命令进入obj目录。

cd obj

步骤6 执行以下命令进行编译安装。

../configure --disable-multilib --enable-languages="c,c++,fortran" --prefix=*/path/to/GNU* **--disable-static --enable-shared --with-gmp=***/path/to/GMP*/

gmp-6.1.0 --with-mpfr=*/path/to/MPFR*/mpfr-3.1.4 --with-mpc=*/path/to/MPC*/mpc-1.0.3

make

make install

步骤7 执行以下命令加载环境变量。

export PATH*=/path/to/GNU*/bin:$PATH

export LD_LIBRARY_PATH*=/path/to/GNU*/lib64:$LD_LIBRARY_PATH

编译安装成功如下图所示。

```
[root@          gcc-9.1.0]# ls
ABOUT-NLS           config.rpath  COPYING.RUNTIME  intl          libgcc        libphobos        ltmain.sh          Makefile.in     obj
ar-lib              config.sub    depcomp          LAST_UPDATED  libgfortran   libquadmath      lt~obsolete.m4     Makefile.tpl    README
ChangeLog           configure     fixincludes      libada        libgo         libsanitizer     lto-plugin         MD5SUMS         symlink-tree
ChangeLog.jit       configure.ac  gcc              libatomic     libgomp       libssp           ltoptions.m4       missing         test-driver
ChangeLog.tree-ssa  contrib       gnattools        libbacktrace  libhsail-rt   libstdc++-v3     ltsugar.m4         mkdep           ylwrap
compile             COPYING       gotools          libcc1        libiberty     libtool-ldflags  ltversion.m4       mkinstalldirs   zlib
config              COPYING3      include          libcpp        libitm        libtool.m4       MAINTAINERS        move-if-change
config.guess        COPYING3.LIB  INSTALL          libdecnumber  libobjc       libvtv           maintainer-scripts multilib.am
config-ml.in        COPYING.LIB   install-sh       libffi        liboffloadmic ltgcc.m4         Makefile.def       NEWS
[root@          gcc-9.1.0]# cd obj/
[root@          obj]# ls
aarch64-unknown-linux-gnu        lib          prev-aarch64-unknown-linux-gnu  serdep.tmp                        stage1-lto-plugin
bin                              lib64        prev-fixincludes                share                             stage1-zlib
build-aarch64-unknown-linux-gnu  libbacktrace prev-gcc                       stage1-aarch64-unknown-linux-gnu  stage_current
compare                          libcc1       prev-intl                       stage1-fixincludes                stage_final
config.log                       libcpp       prev-libbacktrace               stage1-gcc                        stage_last
config.status                    libdecnumber prev-libcpp                     stage1-intl                       zlib
fixincludes                      libexec      prev-libdecnumber               stage1-libbacktrace
gcc                              liberty      prev-liberty                    stage1-libcpp
include                          lto-plugin   prev-lto-plugin                 stage1-libdecnumber
intl                             Makefile     prev-zlib                       stage1-liberty
[root@          obj]# cd bin
[root@          bin]# ls
aarch64-unknown-linux-gnu-c++  aarch64-unknown-linux-gnu-gcc-9.1.0  aarch64-unknown-linux-gnu-gcc-ranlib  cpp  gcc-ar      gcov       gfortran
aarch64-unknown-linux-gnu-g++  aarch64-unknown-linux-gnu-gcc-ar     aarch64-unknown-linux-gnu-gfortran    g++  gcc-nm      gcov-dump
aarch64-unknown-linux-gnu-gcc  aarch64-unknown-linux-gnu-gcc-nm     c++                                   gcc  gcc-ranlib  gcov-tool
[root@          bin]# cd ..
[root@          obj]# cd lib
lib/          lib64/        libbacktrace/ libcc1/       libcpp/       libdecnumber/ libexec/      libiberty/
[root@          obj]# cd lib64
[root@          lib64]# ls
libasan.la          libcc1.la         libgfortran.so.5.0.0  libitm.so.1       libssp.la           libstdc++.so              libtsan.so.0
libasan_preinit.o   libcc1.so         libgfortran.spec      libitm.so.1.0.0   libssp_nonshared.a  libstdc++.so.6            libtsan.so.0.0.0
libasan.so          libcc1.so.0       libgomp.la            libitm.spec       libssp_nonshared.la libstdc++.so.6.0.26       libubsan.la
libasan.so.5        libcc1.so.0.0.0   libgomp.so            liblsan.la        libssp.so           libstdc++.so.6.0.26-gdb.py libubsan.so
libasan.so.5.0.0    libgcc_s.so       libgomp.so.1          liblsan_preinit.o libssp.so.0         libsupc++.a               libubsan.so.1
libatomic.la        libgcc_s.so.1     libgomp.so.1.0.0      liblsan.so        libssp.so.0.0.0     libsupc++.la              libubsan.so.1.0.0
libatomic.so        libgfortran.la    libgomp.spec          liblsan.so.0      libstdc++fs.a       libtsan.la
libatomic.so.1      libgfortran.so    libitm.la             liblsan.so.0.0.0  libstdc++fs.la      libtsan_preinit.o
libatomic.so.1.2.0  libgfortran.so.5  libitm.so             libsanitizer.spec  libstdc++.la       libtsan.so
```

----结束

2.5 验证 GNU9.1

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令加载环境变量。

export PATH=*/path/to/GNU*/bin:$ PATH

export LD_LIBRARY_PATH=*/path/to/GNU*/lib64:$LD_LIBRARY_PATH

步骤3 执行以下命令查看gcc版本是否为9.1.0。

gcc -v

```
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/path/to/GNU/bin/../libexec/gcc/aarch64-unknown-linux-gnu/9.1.0/lto-wrapper
Target: aarch64-unknown-linux-gnu
Configured with: ../configure --disable-multilib --enable-languages=c,c++,fortran --prefix=/path/to/GNU --disable-static --enable-shared --with-gmp=/path/to/GMP --with-mpfr=/path/to/MPFR --with-mpc=/path/to/MPC
```



```
Thread model: posix
gcc version 9.1.0 (GCC)
```

----结束

3 HTCondor 8.9.2 安装指南

3.1 介绍

简要介绍

HTCondor是一个开源的高吞吐量计算软件框架，用于计算密集型任务的粗粒度分布式并行化。它可用于管理专用计算机群集上的工作负载，或将工作分配给空闲的台式计算机，即所谓的循环清理。HTCondor可在Linux，Unix，Mac OS X，FreeBSD和Microsoft Windows操作系统上运行。HTCondor可以将专用资源（机架式集群）和非专用台式机（循环清理）集成到一个计算环境中。

建议的版本

建议使用版本为“HTCondor 8.9.2”。

3.2 环境信息

硬件要求

硬件要求如**表3-1**所示。

表 3-1 硬件要求

项目	说明
CPU	Kunpeng 920

软件要求

软件要求如表3-2所示。

表 3-2 软件要求

项目	版本	下载地址
HTCondor	8.9.2	https://github.com/htcondor/htcondor/releases/tag/V8_9_2
munge	0.5.13	https://github.com/dun/munge/releases/tag/munge-0.5.13

操作系统要求

操作系统要求如表3-3所示。

表 3-3 操作系统要求

项目	版本	下载地址
CentOS	7.6	https://www.centos.org/download/

3.3 部署规划数据

本章节给出HTCondor软件在安装部署过程中涉及到的相关软件安装规划路径的用途及详细说明。

表 3-4 部署规划数据

序号	软件安装规划路径	用途	说明
1	*/path/to/*HTCONDOR	HTCondor的安装规划路径。	这里的安装规划路径只是一个举例说明，建议部署在共享路径中。现网需要根据实际情况调整，后续章节凡是遇到安装路径的命令，都以现网实际

序号	软件安装规划路径	用途	说明
2	*/path/to/MUNGE*	munge的安装规划路径。	规划的安装路径为准进行替换，不再单独说明。

3.4 部署 Htcondor

3.4.1 下载安装包

操作步骤

步骤1 下载Htcondor安装包“htcondor-8_9_2.tar.gz”。

下载地址：**https://github.com/htcondor/htcondor/archive/V8_9_2.tar.gz**

步骤2 下载munge安装包“munge-0.5.13.tar.xz”。

下载地址：**https://github.com/dun/munge/releases/tag/munge-0.5.13**

步骤3 使用SFTP工具：

- 将Htcondor安装包上传至服务器*“/path/to/HTCONDOR”*目录。
- 将Munge安装包上传至服务器*“/path/to/MUNGE”*目录。

----结束

3.4.2 依赖库安装

3.4.2.1 munge 安装

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令构建munge的rpm包。

rpmbuild -tb --clean munge-0.5.13.tar.xz

步骤3 执行以下命令检查是否成功生成rpm包。

ls /root/rpmbuild/RPMS/aarch64/ | grep munge

munge-0.5.13-1.el7.aarch64.rpm

munge-debuginfo-0.5.13-1.el7.aarch64.rpm

munge-devel-0.5.13-1.el7.aarch64.rpm

munge-libs-0.5.13-1.el7.aarch64.rpm

步骤4 在*“/path/to/MUNGE”*目录下创建一个“mungerpm”目录，并将“/root/rpmbuild/RPMS/aarch64/”目录下munge的rpm包拷贝到“/path/to/MUNGE/mungerpm”目录。

mkdir -p mungerpm

cp /root/rpmbuild/RPMS/aarch64/munge* */path/to/MUNGE*/mungerpm -f

步骤5 执行以下命令安装munge的rpm包。

cd */path/to/MUNGE*/mungerpm

yum install -y munge-*

----结束

3.4.2.2 其它依赖包

请根据实际情况，安装其他所需依赖包。

执行以下命令补充安装。

yum install -y *xxx*

说明

xxx为安装包的名称。

3.4.3 编译 HTCondor

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令解压安装包。

cd */path/to/HTCONDOR*

tar -xvf htcondor-8_9_2.tar.gz

步骤3 执行以下命令进入“condor-8.9.2”目录。

cd htcondor-8_9_2

ls

```
build  builder.sh  build-on-linux.sh  CITATION.cff  CMakeLists.txt  configure_redhat  configure_uw  doc
docs  externals  LICENSE-2.0.txt  msconfig  nmi_tools  NOTICE.txt  src  view
```

步骤4 执行以下命令编辑HTCondor的config文件。

cp configure_redhat buildarm.sh

1. **vi buildarm.sh**
2. 按“i”进入编辑模式，添加如下内容。

```
#!/bin/sh

echo "-----------------------------------------------------------------------"
echo "* NOTE: Attempting to configure a Red Hat-esk build"
echo "* which builds against system libs and selectively "
echo "* enables and disables portions of condor"
echo "* If you are unsure, you should run \"cmake .\""
echo "*"
echo "* add -D_DEBUG:BOOL=FALSE to get non-optimized code for debugging"
echo "* Another option would be to run ccmake or cmake-gui"
echo "* and select the options you care to build with"
echo "-----------------------------------------------------------------------"
cmake .. \
-D_DEBUG:BOOL=TRUE \
-DWITH_CREAM:BOOL=FALSE \
-DNO_PHONE_HOME:BOOL=TRUE \
-DHAVE_BACKFILL:BOOL=FALSE \
```



```
-DHAVE_BOINC:BOOL=FALSE \
-DHAVE_KBDD:BOOL=TRUE \
-DHAVE_HIBERNATION:BOOL=TRUE \
-DWANT_CONTRIB:BOOL=ON \
-DWANT_MAN_PAGES:BOOL=TRUE \
-DWANT_FULL_DEPLOYMENT:BOOL=FALSE \
-DWANT_GLEXEC:BOOL=FALSE \
-D_VERBOSE:BOOL=TRUE \
-DBUILDID:STRING=RH_development \
-DWITH_GLOBUS:BOOL=FALSE \
-DWITH_VOMS:BOOL=FALSE \
-DCMAKE_INSTALL_PREFIX:PATH=${PWD}/release_dir "$@"
```

3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤5 执行以下命令进行配置。

./buildarm.sh

```
.....
-- Configuring done
-- Generating done
-- Build files have been written to: /home/htcondor/condor-8.9.2
```

步骤6 执行以下命令进行编译安装。

make -j 64

make install

----结束

3.4.4 配置 HTCondor

⚠ 注意

此文章是在单节点的基础上做配置—在一个节点上做管理、提交任务和执行任务。

操作步骤

步骤1 执行以下命令进入“release_dir”目录。

cd */path/to/HTCONDOR*/**htcondor-8_9_2/release_dir**

步骤2 执行以下命令创建“condor.sh”文件。

1. **vi condor.sh**
2. 按“i”进入编辑模式，添加如下内容。

```
export CONDOR_CONFIG=/path/to/HTCONDOR/htcondor-8_9_2/release_dir/etc/condor_config
export PATH=/path/to/HTCONDOR/htcondor-8_9_2/release_dir/bin:/path/to/HTCONDOR/condor-8.9.2/release_dir/sbin:$PATH
```

3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。
4. 执行以下命令使文件生效。

   source condor.sh

步骤3 执行以下命令进入“release_dir/etc”目录。

cd */path/to/HTCONDOR*/**htcondor-8_9_2/release_dir/etc**

步骤4 执行以下命令创建“condor_config”配置文件。

1. **vi condor_config**
2. 按“i”进入编辑模式，添加如下内容。

```
CONDOR_HOST            = 192.168.47.111
RELEASE_DIR            = /path/to/HTCONDOR/htcondor-8_9_2/release_dir
LOCAL_DIR              = /data/
LOCAL_CONFIG_DIR       = $(LOCAL_DIR)/config
LOCAL_CONFIG_FILE      = $(LOCAL_DIR)/condor_config.local
CONDOR_ADMIN           = root@192.168.47.111
MAIL                   = /usr/bin/mail
ALLOW_ADMINISTRATOR    = $(CONDOR_HOST)
ALLOW_NEGOTITATOR      = $(CONDOR_HOST)
LOCK                   = $(LOG)
CONDOR_IDS             = 2001.2001

use SECURITY : HOST_BASED

LOG                    = $(LOCAL_DIR)/log
SPOOL                  = $(LOCAL_DIR)/spool
BIN                    = $(RELEASE_DIR)/bin
LIB                    = $(RELEASE_DIR)/lib
SBIN                   = $(RELEASE_DIR)/sbin
LIBEXEC                = $(RELEASE_DIR)/libexec
HISTORY                = $(RELEASE_DIR)/history


MASTER_LOG             = $(LOG)/MasterLog
SCHEDD_LOG             = $(LOG)/SchedLog
SHADOW_LOG             = $(LOG)/ShadowLog

SHADOW_LOCK            = $(LOCK)/ShadowLock

DAEMON_LIST = COLLECTOR MASTER NEGOTIATOR SCHEDD STARTD
CONDOR_HOST = $(CONDOR_HOST)
USE_CLONE_TO_CREATE_PROCESSES = False
```

3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤5 执行以下命令创建一个condor用户和组。

groupadd -g 2001 condor

useradd -u 2001 -g 2001 condor

步骤6 执行以下命令创建HTCondor所需的目录和文件。

mkdir -p /data

cd /data

mkdir -p config examples execute log spool

touch condor_config.local

touch log/MasterLog log/SchedLog log/ShadowLog log/ShadowLock

chown -R condor.condor *

步骤7 执行以下命令配置condor的init.d服务。

cp */path/to/HTCONDOR*/**htcondor-8_9_2/release_dir/etc/init.d/condor /etc/init.d/ -f**

1. **vi /etc/init.d/condor**
2. 按“i”进入编辑模式，添加如下内容。


```
......
# Path to your primary condor configuration file.
CONDOR_CONFIG="/path/to/HTCONDOR/htcondor-8_9_2/release_dir/etc/condor_config"

# Path to condor_config_val
CONDOR_CONFIG_VAL="/path/to/HTCONDOR/htcondor-8_9_2/release_dir/bin/condor_config_val"
......
```

3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

----结束

3.4.5 启动 HTCondor

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令启动HTCondor。

/etc/init.d/condor start

----结束

3.5 HTCondor 集群配置

3.5.1 配置前注意事项

操作步骤

步骤1 确保节点间通信正常且ssh互信（建议同网段广播通信）。

步骤2 各节点已根据**3.4 部署Htcondor**安装好单节点condor。

步骤3 确定各节点角色（由condor_config文件中DAEMON_LIST参数配置）。

步骤4 检查condor_config文件的权限设置是否正确，用户根据实际需求配置去编辑文件中参数。

----结束

3.5.2 condor_config 文件配置

操作步骤

步骤1 Manager节点“condor_config”文件信息如下：

```
CONDOR_HOST            = 192.168.47.111
RELEASE_DIR            = /path/to/HTCONDOR/htcondor-8_9_2/release_dir
LOCAL_DIR              = /data
LOCAL_CONFIG_DIR       = $(LOCAL_DIR)/config
LOCAL_CONFIG_FILE      = $(LOCAL_DIR)/condor_config.local
CONDOR_ADMIN           = root@192.168.47.111
MAIL                   = /usr/bin/mail
CONDOR_IDS             = 2001.2001

ALLOW_WRITE            = 192.168.47.42
ALLOW_READ             = 192.168.47.42
ALLOW_ADVERTISE_SCHEDD = 192.168.47.*
```

```
ALLOW_ADVERTISE_MASTER = 192.168.47.42
ALLOW_ADVERTISE_STARTD = 192.168.47.42

use SECURITY : HOST_BASED

LOG                 = $(LOCAL_DIR)/log
ALL_DEBUG             = D_ALL
SPOOL               = $(LOCAL_DIR)/spool
LOCK                = $(LOG)
BIN                 = $(RELEASE_DIR)/bin
LIB                 = $(RELEASE_DIR)/lib
SBIN                = $(RELEASE_DIR)/sbin
LIBEXEC             = $(RELEASE_DIR)/libexec
HISTORY             = $(RELEASE_DIR)/history

MASTER_LOG          = $(LOG)/MasterLog
SCHEDD_LOG          = $(LOG)/SchedLog
SHADOW_LOG          = $(LOG)/ShadowLog

SHADOW_LOCK         = $(LOCK)/ShadowLock

DAEMON_LIST = COLLECTOR MASTER NEGOTIATOR SCHEDD STARTD
USE_CLONE_TO_CREATE_PROCESSES = False
```

步骤2 提交或执行节点condor_config文件信息如下：

```
CONDOR_HOST         = 192.168.47.111
RELEASE_DIR         = /path/to/HTCONDOR/htcondor-8_9_2/release_dir
LOCAL_DIR           = /data
LOCAL_CONFIG_DIR    = $(LOCAL_DIR)/config
LOCAL_CONFIG_FILE   = $(LOCAL_DIR)/condor_config.local
CONDOR_ADMIN        = root@192.168.47.111
MAIL                = /usr/bin/mail
CONDOR_IDS          = 2008.2008

ALLOW_WRITE         = 192.168.47.*
ALLOW_READ          = 192.168.47.*

use SECURITY : HOST_BASED
LOG                 = $(LOCAL_DIR)/log
SPOOL               = $(LOCAL_DIR)/spool
LOCK                = $(LOCAL_DIR)/lock
BIN                 = $(RELEASE_DIR)/bin
LIB                 = $(RELEASE_DIR)/lib
SBIN                = $(RELEASE_DIR)/sbin
LIBEXEC             = $(RELEASE_DIR)/libexec
HISTORY             = $(RELEASE_DIR)/history

MASTER_LOG          = $(LOG)/MasterLog
SCHEDD_LOG          = $(LOG)/SchedLog
SHADOW_LOG          = $(LOG)/ShadowLog
SHADOW_LOCK         = $(LOCK)/ShadowLock

DAEMON_LIST = MASTER STARTD SCHEDD
USE_CLONE_TO_CREATE_PROCESSES = False
```

步骤3 执行以下命令重新插入“condor_config”配置（注：管理及作业节点都需要执行）。

condor_reconfig

步骤4 执行以下命令查看condor队列状态（正常为节点核数总和）。

condor_status

结果样例如下所示。


```
               Machines Owner Claimed Unclaimed Matched Preempting  Drain

aarch64/LINUX       224     0       0       224       0          0      0

        Total       224     0       0       224       0          0      0
```

----结束

3.6 验证 Htcondor

3.6.1 常用命令

表 3-5 常用命令

命令	说明
condor_q	查看任务队列
condor_status	查看资源状态
condor_history	查看历史任务
condor_submit	提交作业
congdor_rm	删除作业

3.6.2 提交脚本作业

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令进入“condor”用户。

su - condor

步骤3 执行以下命令进入“/data/examples”目录。

cd /data/examples

步骤4 执行以下命令编写“test.sh”文件。

1. **vi test.sh**
2. 按“i”进入编辑模式，添加如下内容。

```
#!/bin/bash
for x in {0..10}
do
echo "This is a test"
sleep 5s
done
```

3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤5 执行以下命令编写“test.sub”文件。

1. **vi test.sub**
2. 按“i”进入编辑模式，添加如下内容。

```
universe         = vanilla
executable       = ./test.sh
output           = test.o
error            = test.e
log              = test.log
should_transfer_files   = YES
when_to_transfer_output = ON_EXIT
Notification             = never
queue
```

3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤6 执行以下命令提交作业。

condor_submit test.sub

步骤7 执行以下命令提交多队列作业，验证集群。

cat test.sub

```
universe         = vanilla
executable       = ./test.sh
output           = test.o
error            = test.e
log              = test.log
should_transfer_files   = YES
when_to_transfer_output = ON_EXIT
Notification             = never
queue 150
```

步骤8 执行以下命令进行验证。

condor_submit test.sub


```
slot74@TsXA320V2 LINUX      aarch64 Claimed   Busy      0.000 2710  0+00:00:19
slot75@TsXA320V2 LINUX      aarch64 Claimed   Busy      0.000 2710  0+00:00:19
slot76@TsXA320V2 LINUX      aarch64 Claimed   Busy      0.000 2710  0+00:00:20
slot77@TsXA320V2 LINUX      aarch64 Claimed   Busy      0.000 2710  0+00:00:19
slot78@TsXA320V2 LINUX      aarch64 Claimed   Busy      0.000 2710  0+00:00:19
slot79@TsXA320V2 LINUX      aarch64 Claimed   Busy      0.000 2710  0+00:00:19
slot80@TsXA320V2 LINUX      aarch64 Claimed   Busy      0.010 2710  0+00:00:09
slot81@TsXA320V2 LINUX      aarch64 Claimed   Busy      0.010 2710  0+00:00:09
slot82@TsXA320V2 LINUX      aarch64 Claimed   Busy      0.020 2710  0+00:00:09
slot83@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:05
slot84@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:05
slot85@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:04
slot86@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:04
slot87@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:04
slot88@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:03
slot89@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:03
slot90@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:03
slot91@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:02
slot92@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:02
slot93@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:01
slot94@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:01
slot95@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:01
slot96@TsXA320V2 LINUX      aarch64 Unclaimed Idle      0.000 2710  0+00:00:00
slot1@Ts2280V2   LINUX      aarch64 Claimed   Busy      0.010 4078  0+00:00:02
slot2@Ts2280V2   LINUX      aarch64 Claimed   Busy      0.000 4078  0+00:00:02
slot3@Ts2280V2   LINUX      aarch64 Claimed   Busy      0.000 4078  0+00:00:02
slot4@Ts2280V2   LINUX      aarch64 Claimed   Busy      0.000 4078  0+00:00:03
slot5@Ts2280V2   LINUX      aarch64 Claimed   Busy      0.000 4078  0+00:00:03
slot6@Ts2280V2   LINUX      aarch64 Claimed   Busy      0.000 4078  0+00:00:02
slot7@Ts2280V2   LINUX      aarch64 Claimed   Busy      0.000 4078  0+00:00:04
slot8@Ts2280V2   LINUX      aarch64 Claimed   Busy      0.000 4078  0+00:00:04
```

```
               Machines Owner Claimed Unclaimed Matched Preempting  Drain

aarch64/LINUX       224     0     136        88       0          0      0

        Total       224     0     136        88       0          0      0
```

----结束

3.7 故障排除

3.7.1 解决 htcondor 在鲲鹏 920 上启动服务失败

问题描述：

HTCondor在鲲鹏920上启动服务失败。

问题处理：

步骤1 执行以下命令编辑“daemon_core.cpp”文件。

1. **vi condor-8.9.2/src/condor_daemon_core.V6/daemon_core.cpp**
2. 按“i”进入编辑模式，编辑第5856-5880行内容。

```
if( daemonCore->UseCloneToCreateProcesses() ) {
dprintf(D_FULLDEBUG,"Create_Process: using fast clone() "
"to create child process.\n");

// The stack size must be big enough for everything that
// happens in CreateProcessForkit::clone_fn().  In some
// environments, some extra steps may need to be taken to
// make a stack on the heap (to mark it as executable), so
// we just do it using the parent's stack space and we use
// CLONE_VFORK to ensure the child is done with the stack
// before the parent throws it away.
//const int stack_size = 16384;
const int stack_size = 64*1024*2;
```



```
//const int stack_size = 64*1024*16;
char child_stack[stack_size] ;

// Beginning of stack is at end on all processors that run
// Linux, except for HP PA.  Here we just detect at run-time
// which way it goes.
char *child_stack_ptr = child_stack;
if( stack_direction() == STACK_GROWS_DOWN ) {
child_stack_ptr += stack_size;
}
child_stack_ptr = (char *)((std::uintptr_t)child_stack_ptr & ~(std::uintptr_t)0<<4);
```

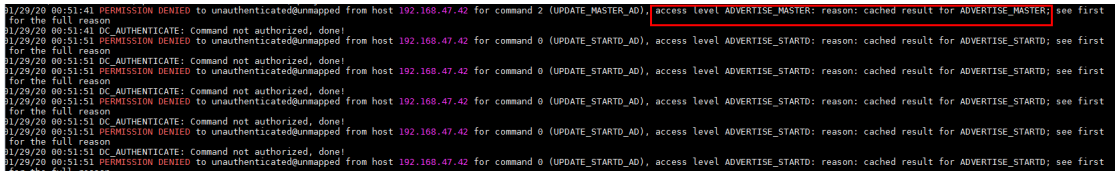
3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

----结束

3.7.2 权限造成的连接被拒绝问题

问题描述：

权限造成的连接被拒绝问题。



```
01/29/20 00:51:41 PERMISSION DENIED to unauthenticated@unmapped from host 192.168.47.42 for command 2 (UPDATE_MASTER_AD), access level ADVERTISE_MASTER: reason: cached result for ADVERTISE_MASTER; see first
 for the full reason
01/29/20 00:51:41 DC_AUTHENTICATE: Command not authorized, done!
01/29/20 00:51:51 PERMISSION DENIED to unauthenticated@unmapped from host 192.168.47.42 for command 0 (UPDATE_STARTD_AD), access level ADVERTISE_STARTD: reason: cached result for ADVERTISE_STARTD; see first
 for the full reason
01/29/20 00:51:51 DC_AUTHENTICATE: Command not authorized, done!
01/29/20 00:51:51 PERMISSION DENIED to unauthenticated@unmapped from host 192.168.47.42 for command 0 (UPDATE_STARTD_AD), access level ADVERTISE_STARTD: reason: cached result for ADVERTISE_STARTD; see first
 for the full reason
01/29/20 00:51:51 DC_AUTHENTICATE: Command not authorized, done!
01/29/20 00:51:51 PERMISSION DENIED to unauthenticated@unmapped from host 192.168.47.42 for command 0 (UPDATE_STARTD_AD), access level ADVERTISE_STARTD: reason: cached result for ADVERTISE_STARTD; see first
 for the full reason
01/29/20 00:51:51 DC_AUTHENTICATE: Command not authorized, done!
01/29/20 00:51:51 PERMISSION DENIED to unauthenticated@unmapped from host 192.168.47.42 for command 0 (UPDATE_STARTD_AD), access level ADVERTISE_STARTD: reason: cached result for ADVERTISE_STARTD; see first
 for the full reason
01/29/20 00:51:51 DC_AUTHENTICATE: Command not authorized, done!
01/29/20 00:51:51 PERMISSION DENIED to unauthenticated@unmapped from host 192.168.47.42 for command 0 (UPDATE_STARTD_AD), access level ADVERTISE_STARTD: reason: cached result for ADVERTISE_STARTD; see first
```

问题原因：

因为ADVERTISE_MASTER守护权限不足，需要在condor_config文件配置ALLOW_ADVERTISE_MASTER = 192.168.47.42，该参数支持子网匹配如192.168.47.*。

问题处理：

- 日志文件如果没有什么特别的配置，通常只会打印D_ALWAYS级别的信息，如果需要深入查看各服务的日志信息，可在“condor_config”文件中添加如下信息：

  ```
  ALL_DEBUG   =  D_ALL
  ```

- condor-8.9.*以上版本“condor_config”文件中只配置“ALLOW_WRITE”、“ALLOW_READ”是不够的，还需要“加上use SECURITY : HOST_BASED”。
- 权限错误则根据完整的日志报错配置“condor_config”相关参数；如ADVERTISE_ SCHEDD权限不足，则需在“condor_config”文件中添加如下信息：

  ```
  ALLOW_ADVERTISE_ SCHEDD = 192.168.47.*
  ```

- “condor_config”配置参数的等号间需要空格隔开，否则不生效，如下所示：

  ```
  ALLOW_WRITE  =  192.168.47.42, 192.168.47.111
  ```

3.8 更多资源

1. HTCondor官网安装指导：**http://research.cs.wisc.edu/htcondor/**
2. 入门多节点：**https://spinningmatt.wordpress.com/2011/06/12/getting-started-creating-a-multiple-node-condor-pool/**
3. Need host-based security at least for HTCondor 8.9+ ：**https://github.com/opensciencegrid/condor-cron/pull/10/files#diff-9fab5b60a4551556ee9b100bb56030ee**

4. condor问题定位邮件网址：**https://www-auth.cs.wisc.edu/lists/htcondor-users/2018-October/threads.shtml**

4 Lustre 2.12.2 安装指南

4.1 介绍

简要介绍

Lustre，一种并行文件系统，通常用于大型计算机集群和超级计算机。Lustre是源自Linux和Cluster的混成词。最早在1999年，由皮特 · 布拉姆创建的集群文件系统公司开始研发，于2003年发布Lustre 1.0，采用GNU GPLv2开源码授权。

在HPC解决方案中，存储方案是关键环节之一，而Lustre并行文件系统则是HPC存储方案中很常见的一种解决方案，随着Kunpeng 920平台的推出，需要尽快验证客户端在其的可用性与性能。此文档主要是指导如何在Kunpeng 920平台上基于源码对客户端重新进行编译。

建议的版本

建议使用版本为“Lustre 2.12.2”。

4.2 环境信息

硬件要求

硬件要求如**表4-1**所示。

表 4-1 硬件要求

项目	说明
CPU	Kunpeng 920

软件要求

软件要求如表4-2所示。

表 4-2 软件要求

项目	版本	下载地址
开源Lustre客户端源码包	2.12.2	https://downloads.whamcloud.com/public/lustre/lustre-2.12.2/el7.6.1810/client/SRPMS/

操作系统要求

操作系统要求如表4-3所示。

表 4-3 操作系统要求

项目	版本	下载地址
CentOS	7.6	https://www.centos.org/download/

4.3 部署 Lustre

4.3.1 客户端编译

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令安装源码包。

cd /tmp

rpm -ivh lustre-2.12.2-1.src.rpm lustre-client-dkms-2.12.2-1.el7.src.rpm

步骤3 执行以下命令进入安装目录。

cd /root/rpmbuild/SOURCES

步骤4 执行以下命令解压源码包。

tar -xvf lustre-2.12.2.tar.gz

步骤5 执行以下命令进行安装目录。

cd lustre-2.12.2

步骤6 执行以下命令进行安装编译。

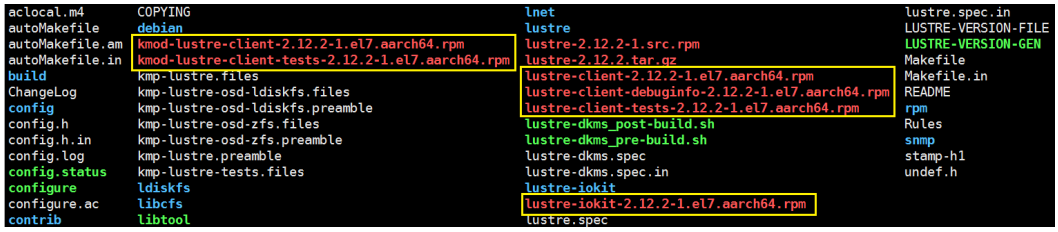
./configure --with-o2ib=/usr/src/ofa_kernel/default/

make rpms

步骤7 编译成功后执行以下命令查看在目录下生产相关的rpm包。

ls

结果样例如下所示。



----结束

4.3.2 客户端安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令进入安装目录。

cd /root/rpmbuild/SOURCES/lustre-2.12.2

步骤3 执行以下命令安装相关客户端的RPM包。

rpm -ivh --nodeps kmod-lustre-client*.rpm lustre-client*.rpm lustre-iokit*.rpm

----结束

4.4 配置 Lustre

4.4.1 客户端 LNET 配置

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令配置客户端LNET。

lustre_rmmod

modprobe lnet

lctl network down

步骤3 执行以下命令编译文件。

1. **vi /etc/modprobe.d/iml_lnet_module_parameters.conf**
2. 按“i”进入编辑模式，添加如下内容。
   ```
   options lnet networks=o2ib0(ib0)
   ```
3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤4 执行以下命令继续配置客户端LNET。

modprobe lustre

lctl network up

步骤5 执行以下命令检查客户端LNET配置。

lctl list_nids

```
10.10.10.101@o2ib
```

步骤6 使用LNET ping工具检测与Lustre服务端的LNET间通信状态。

lctl ping *10.10.10.105@o2ib0*

```
12345-0@lo
12345-10.10.10.105@o2ib
```

----结束

4.4.2 文件系统挂载

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令创建目录并挂载。

mkdir /mnt/lustre

mount -t lustre 10.10.10.105@o2ib0 :/lustre /mnt/lustre

----结束

4.4.3 文件系统基础功能认验证

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令验证基础写操作。

echo “This is the basic write test for lustre” >> /mnt/lustre/test.log

步骤3 执行以下命令验证基础读操作。

cat /mnt/lustre/test.log

```
This is the basic write test for lustre
```

----结束

5 OpenHPC 1.3.8 安装指南

5.1 介绍

简要介绍

OpenHPC是一套基于Linux的HPC的社区驱动的FOSS（Free and open source software，自由及开放源代码软件）工具。OpenHPC对硬件没有特殊要求。

OpenHPC提供了一个集成且经过测试的软件组件集合，以及支持的标准Linux发行版，可用于实现功能齐全的计算集群。组件涵盖整个HPC软件生态系统，包括配置和系统管理工具，资源管理，I/O服务，开发工具，数值库和性能分析工具。

OpenHPC官方网站是**https://openhpc.community**，源码托管在github上**https://github.com/openhpc/ohpc**。

建议的版本

建议使用版本为“OpenHPC 1.3.8”。

5.2 环境信息

硬件要求

硬件要求如**表5-1**所示。

表 5-1 硬件要求

项目	说明
CPU	Kunpeng 920

软件要求

软件要求如表5-2所示。

表 5-2 软件要求

项目	版本	下载地址
OpenHPC套件	1.3.8	http://build.openhpc.community/dist/1.3.8/OpenHPC-1.3.8.CentOS_7.aarch64.tar

操作系统要求

操作系统要求如表5-3所示。

表 5-3 操作系统要求

项目	版本	下载地址
CentOS	7.6	https://www.centos.org/download/

5.2.1 基础软件堆栈

类别	支持的软件	版本号	获取地址
编译器	GNU	4.9	CentOS自带
	GNU	5	https://mirrors.ustc.edu.cn/gnu/
			OpenHPC 1.3.8包含
	GNU	6	https://mirrors.ustc.edu.cn/gnu/
	GNU	7	https://mirrors.ustc.edu.cn/gnu/
			OpenHPC 1.3.8包含
	GNU	8	https://mirrors.ustc.edu.cn/gnu/
			OpenHPC 1.3.8包含

类别	支持的软件	版本号	获取地址
	GNU	9	https://mirrors.ustc.edu.cn/gnu/
	LLVM	4	OpenHPC 1.3.8包含
	LLVM	5	OpenHPC 1.3.8包含
	R	3.3	OpenHPC 1.3.8包含
MPI	HPC-X	2.4	http://www.mellanox.com/downloads/hpc/hpc-x/v2.4/hpcx-v2.4.0-gcc-MLNX_OFED_LINUX-4.6-1.0.1.1-redhat7.6-aarch64.tbz
	OpenMPI	3.1	https://www.open-mpi.org/software/ompi/v3.1
			OpenHPC 1.3.8包含
		4.0.1	https://www.open-mpi.org/software/ompi/v4.0
	mpich	3.3	OpenHPC 1.3.8包含
	mvapich	2.3	CentOS自带
数学&IO库	OpenBLAS	0.2.19	OpenHPC 1.3.8包含
	Scalapack	2	OpenHPC 1.3.8包含
	FFTW	3.3	OpenHPC 1.3.8包含
	SuperLU_Dist	4.2	OpenHPC 1.3.8包含
	MUMPS	5	OpenHPC 1.3.8包含
	hdf5	1.8.17	OpenHPC 1.3.8包含
	netcdf	4.4	OpenHPC 1.3.8包含
	parallel-netcdf	1.8	OpenHPC 1.3.8包含
		1.9	
		1.11	
	petsc	3.7	OpenHPC 1.3.8包含
	boost	1.61	OpenHPC 1.3.8包含
		1.63	
	gsl	2.4	OpenHPC 1.3.8包含
		2.5	
	hypre	2.13	OpenHPC 1.3.8包含

类别	支持的软件	版本号	获取地址
		2.14	
		2.15	
调度软件	Slurm	18.08.7	OpenHPC 1.3.8包含
	Open PBS Pro	19.1.1	OpenHPC 1.3.8包含
管理软件	联科CEHSS	5.3	**https://www.clustertech.com/zh-hans/**高性能计算
	warewulf	1.3.8	OpenHPC 1.3.8包含
存储软件	Lustre Client	2.1.2	**https://downloads.whamcloud.com/public/lustre/lustre-2.12.2/el7.6.1810/client/SRPMS/**
	BeeGFS	7.1.3	**https://git.beegfs.io/pub/v7/tree/7.1.3**
	NFS	N/A	CentOS自带
Benchmark	HPL	2.3	**https://www.netlib.org/benchmark/hpl/hpl-2.3.tar.gz**
	HPCG	N/A	**https://github.com/hpcg-benchmark/hpcg**
	stream	N/A	**https://www.cs.virginia.edu/stream/FTP/Code/**
	IOR	3.1	**https://github.com/hpc/ior**
	mdtest	3.1	**https://github.com/hpc/ior**
	IMB	4.1	OpenHPC 1.3.8包含
	osu mpi benchmark	5.6.1	OpenHPC 1.3.8包含
monitor	ganglia	3.7.2	OpenHPC 1.3.8包含
	nagios	2.1.1	OpenHPC 1.3.8包含

5.3 部署规划数据

本章节给出OpenHPC软件在安装部署过程中涉及到的相关软件安装规划路径的用途及详细说明。

表 5-4 部署规划数据

序号	软件安装规划路径	用途	说明
1	*/path/to/OPENHPC*	OpenHPC的安装规划路径。	这里的安装规划路径只是一个举例说明，建议部署在共享路径中。现网需要根据实际情况调整，后续章节凡是遇到安装路径的命令，都以现网实际规划的安装路径为准进行替换，不再单独说明。

5.4 部署 OpenHPC

5.4.1 下载安装包

操作步骤

步骤1 下载OpenHPC安装包“OpenHPC-1.3.8.CentOS_7.aarch64.tar”。

下载地址：**http://build.openhpc.community/dist/1.3.8/OpenHPC-1.3.8.CentOS_7.aarch64.tar**

步骤2 使用SFTP工具将OpenHPC安装包上传至服务器“*/path/to/OPENHPC*”目录。

----结束

5.4.2 OpenHPC 单节点部署

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令解压安装包。

cd */path/to/OPENHPC*

tar -xvf OpenHPC-1.3.8.CentOS_7.aarch64.tar

步骤3 执行以下命令查看文件。

ls

```
CentOS_7  make_repo.sh  OpenHPC-1.3.8.CentOS_7.aarch64.tar  OpenHPC.local.repo  READM
```

步骤4 执行以下命令执行“make_repo.sh”文件。

./make_repo.sh

结果样例如下所示。

```
Creating OpenHPC.local.repo file in /etc/yum.repos.d
Local repodata stored in /opt/OpenHPC
```

----结束

5.4.3 OpenHPC 在集群下的部署

操作步骤

步骤1 所有节点上传OpenHPC原安装包到本地*/path/to/OPENHPC*目录下。

说明

*/path/to/OPENHPC*目录不要设置NFS共享。

步骤2 执行以下命令集群下使用clush批处理命令,批量解压压缩包。

clush -a “tar -xvf */path/to/OPENHPC/*OpenHPC-1.3.8.CentOS_7.aarch64.tar”

步骤3 执行以下命令批量执行解压出的make_repo.sh文件。

clush -a “cd */path/to/OPENHPC/*OpenHPC-1.3.8.CentOS_7.aarch64;./make_repo.sh”

----结束

5.4.4 OpenHPC 基础环境加载

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令使用yum安装“lmod-ohpc”。

yum install lmod-ohpc

步骤3 执行以下命令使其生效。

source /etc/profile.d/lmod.sh

步骤4 执行以下命令修改“/root/.bashrc”文件。

1. **vi /root/.bashrc**
2. 按“i”进入编辑模式，修改如下内容。
```
module use /opt/ohpc/pub/modulefiles/
module use /opt/ohpc/pub/moduledeps/gnu8/
module use /opt/ohpc/pub/moduledeps/gnu8-openmpi3/
```
3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤5 执行以下命令使环境变量生效。

source /root/.bashrc

----结束

5.5 组件安装

5.5.1 作业调度系统安装

5.5.1.1 Slurm 安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令安装Slurm服务端。

yum install -y slurm-ohpc slurm-slurmctld-ohpc slurm-slurmdbd-ohpc

步骤3 执行以下命令启动服务。

1. **vi /etc/slrum.conf**
2. 按“i”进入编辑模式，添加如下内容。
   ```
   systemctl start slurmctld
   systemctl enable slurmctld
   ```
3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤4 执行以下命令安装slurm客户端。

yum install -y slurm-slurmd-ohpc

步骤5 执行以下命令启动服务。

1. **vi /etc/slurm.conf**
2. 按“i”进入编辑模式，添加如下内容。
   ```
   systemctl enable slurmd
   systemctl start slurmd
   ```
3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤6 Slurm配置参考**《Slurm 18.08.7 安装指南》**中的“安装Slurm”。

----结束

5.5.2 编译器安装

5.5.2.1 GNU 编译器安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 在管理节点上安装部署OpenHPC套件自带的GNU编译器。

yum install -y gnu8-compilers-ohpc

module add gnu8/8.3.0

----结束

5.5.3 数学库安装

5.5.3.1 NetCDF 库安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 在管理节点上安装部署NetCDF库。

yum install -y netcdf-gnu8-openmpi3-ohpc

module add netcdf/4.6.3

----结束

5.5.3.2 NetCDF-Fortra 库安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 在管理节点上安装部署NetCDF-Fortra库。

yum install -y netcdf-fortran-gnu8-openmpi3-ohpc

module add netcdf-fortran/4.4.5

----结束

5.5.3.3 PnetCDF 库安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 在管理节点上安装部署PnetCDF库。

yum install -y pnetcdf-gnu8-openmpi3-ohpc

module add pnetcdf/1.11.1

----结束

5.5.3.4 OpenBLAS 库安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 在管理节点上安装部署OpenBLAS库。

module add gnu8/8.3.0

yum install -y openblas-gnu8-ohpc.aarch64

module add openblas/0.3.5

----结束

5.5.3.5 FFTW 库安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 在管理节点上安装部署FFTW库。

module add gnu8/8.3.0

yum install -y fftw-gnu8-openmpi3-ohpc.aarch64

module add fftw/3.3.8

----结束

5.5.3.6 Metis 库安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 在管理节点上安装部署Metis库。

module add gnu8/8.3.0

yum install -y metis-gnu8-ohpc.aarch64

module add metis/5.1.0

----结束

5.5.3.7 SuperLU 库安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 在管理节点上安装部署SuperLU库。

module add gnu8/8.3.0

yum install -y superlu-gnu8-ohpc.aarch64

module add superlu/5.2.1

----结束

5.5.3.8 SuperLU Dist 库安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 在管理节点上安装部署SuperLU Dist库。

module add gnu8/8.3.0

yum install -y superlu_dist-gnu8-openmpi3-ohpc.aarch64

module add superlu_dist/6.1.1

module add openmpi3/3.1.4

----结束

5.5.3.9 ScaLAPACK 库安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 在管理节点上安装部署ScaLAPACK库。

module add gnu8/8.3.0

yum install -y scalapack-gnu8-openmpi3-ohpc.aarch64

module add superlu_dist/6.1.1

module add scalapack/2.0.2

----结束

5.5.3.10 Hwloc 库安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 在管理节点上安装部署Hwloc库。

yum install hwloc-ohpc.aarch64 -y

module add hwloc/2.0.3

export PATH=/opt/ohpc/pub/libs/hwloc/2.0.3/include:$PATH

----结束

5.5.3.11 Spack 库安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 在管理节点上安装部署Spack库。

yum install -y spack-ohpc.noarch

module add spack/0.12.1

----结束

5.5.4 容器安装

5.5.4.1 Singularity 容器安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令部署Singularity容器及配置环境变量。

yum install -y singularity-ohpc.aarch64

module add singularity/3.2.1

步骤3 执行以下命令外网下载镜像。

singularity pull library://sylabsed/examples/lolcow

----结束

5.5.5 资源监控软件安装

5.5.5.1 Ganglia 监控软件安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令在外网下载安装依赖包。

wget http://www.rpmfind.net/linux/epel/7/aarch64/Packages/l/libconfuse-2.7-7.el7.aarch64.rpm

wget http://www.rpmfind.net/linux/remi/enterprise/7/remi/aarch64/php-ZendFramework-1.12.20-1.el7.remi.noarch.rpm

步骤3 执行以下命令安装以上安装包。

yum install -y libconfuse-2.7-7.el7.aarch64.rpm php-ZendFramework-1.12.20-1.el7.remi.noarch.rpm

步骤4 执行以下命令在服务端安装Ganglia包。

yum install -y ohpc-ganglia

步骤5 执行以下命令在客户端安装Ganglia包。

yum install -y ganglia-gmond-ohpc

步骤6 执行以下命令在服务端配置Ganglia。

perl -pi -e “s/<sms>/${sms_name}/” /etc/ganglia/gmond.conf

说明

将gmond.conf配置文件的HOST的内容修改为服务端的主机名或者IP地址。

步骤7 执行以下命令在客户端上配置Ganglia。

scp SMS_IP:/etc/ganglia/gmond.conf /etc/ganglia

echo “gridname Mysite” >> /etc/ganglia/gmetad.conf

步骤8 执行以下命令在客户端上启动相关进程。

systemctl start gmond

systemctl enable gmond

步骤9 执行以下命令在服务端上启动相关进程。

systemctl start gmond

systemctl start gmetad

systemctl enable gmond

systemctl enable gmetad

systemctl try-restart httpd

systemctl enable httpd

----结束

5.5.5.2 Nagios 和 NRPE 监控软件的安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令在外网下载安装依赖包。

wget http://www.rpmfind.net/linux/fedora/linux/releases/32/Everything/aarch64/os/Packages/q/qstat-2.15-11.20200131gitd1469ab.fc32.aarch64.rpm

wget http://www.rpmfind.net/linux/epel/7/aarch64/Packages/f/fping-3.10-4.el7.aarch64.rpm

wget http://www.rpmfind.net/linux/epel/7/ppc64/Packages/p/python2-mock-1.0.1-10.el7.noarch.rpm

步骤3 执行以下命令安装以上安装包。

yum install qstat-2.15-7.20150619gita60436.fc29.aarch64.rpm fping-3.10-4.el7.aarch64.rpm python2-mock-1.0.1-10.el7.noarch.rp

步骤4 执行以下命令在管理节点上安装Nagios。

yum install ohpc-nagios -y

步骤5 执行以下命令在计算节点上安装NRPE。

yum install nagios-plugins-all-ohpc nrpe-ohpc -y

步骤6 执行以下命令在计算节点上使能和配置NRPE。

systemctl enable nrpe

perl -pi -e "s/^allowed_hosts=/# allowed_hosts=/" /etc/nagios/nrpe.cfg

echo "nrpe 5666/tcp # NRPE" >> /etc/services

echo "nrpe : *192.168.47.111* : ALLOW" >> /etc/hosts.allow

echo "nrpe : ALL : DENY" >> /etc/hosts.allow

/usr/sbin/useradd -c "NRPE user for the NRPE service" -d /var/run/nrpe -r -g nrpe -s /sbin/nologin nrpe

/usr/sbin/groupadd -r nrpe

说明

192.168.47.111是Nagios服务端的IP地址。

步骤7 执行以下命令在计算节点上配置Remote Services。

mv /etc/nagios/conf.d/services.cfg.example /etc/nagios/conf.d/services.cfg

步骤8 执行以下命令在服务端上配置计算节点的内容。

mv /etc/nagios/conf.d/hosts.cfg.example /etc/nagios/conf.d/hosts.cfg

步骤9 执行以下命令在服务端上更新告警的邮件通知信息。

perl -pi -e "s/ \/bin\/mail/ \/usr\/bin\/mailx/g" /etc/nagios/objects/commands.cfg

perl -pi -e "s/nagios\@localhost/root\@${sms_name}/" /etc/nagios/objects/contacts.cfg

步骤10 执行以下命令在服务端上设置webservice用户的密码。

htpasswd -bc /etc/nagios/passwd nagiosadmin huawei

步骤11 执行以下命令在服务端上启动相关进程。

systemctl enable nagios.service

systemctl start nagios.service

chmod u+s `which ping`

systemctl start httpd

systemctl enable httpd

----结束

5.5.6 测试工具安装

5.5.6.1 Performance tools-imb 测试工具安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令在计算节点上安装OpenMPI和IMB的安装包。

yum install openmpi3-gnu7-ohpc.aarch64 -y

yum install imb-gnu7-openmpi3-ohpc.aarch64 -y

步骤3 执行以命令在计算节点的“.bashrc”文件中配置环境变量。

1. **vi /root/.bashrc**
2. 按“i”进入编辑模式，编辑如下内容。

```
module use /opt/ohpc/pub/moduledeps/gnu7-openmpi3/
module use /opt/ohpc/pub/moduledeps/gnu7
module use /opt/ohpc/pub/modulefiles/
module add openmpi3/3.1.0
module add imb/2018.1
```

3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

----结束

5.5.6.2 TAU 测试工具的安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令在计算节点上安装TAU工具。

yum install tau-gnu7-openmpi3-ohpc.aarch64 -y

步骤3 执行以下命令在计算节点上设置环境变量。

module use /opt/ohpc/pub/moduledeps/gnu7-openmpi3/

module use /opt/ohpc/pub/moduledeps/gnu7

module add openmpi3/3.1.0

module add tau/2.27.1

module add imb/2018.1

----结束

5.5.6.3 PAPi 测试工具的安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令在服务器上安装PAPi和Automake。

yum install -y papi-ohpc

yum install -y automake-ohpc

步骤3 执行以下命令设置PAPi和Automake的环境变量。

module add papi/5.7.0

export PATH= /opt/ohpc/pub/utils/autotools/bin:$PATH

----结束

6 OpenMPI 4.0.1 安装指南

6.1 介绍

简要介绍

OpenMPI是一种高性能消息传递库，最初是根据其他几个项目（FT- MPI, LA-MPI, LAM/MPI, 以及 PACX-MPI）来融合的技术和资源，它是MPI-2标准的一个开源实现，由一些科研机构和企业一起开发和维护。因此，OpenMPI能够从高性能社区中获得专业技术、工业技术和资源支持来创建最好的MPI库，提供给系统和软件供应商、程序开发者和研究人员很多便利。

建议的版本

建议使用版本为“OpenMPI 4.0.1”。

6.2 环境要求

硬件要求

硬件要求如**表6-1**所示。

表 6-1 硬件要求

项目	说明
CPU	Kunpeng 920

软件要求

软件要求如表2-2所示。

表 6-2 软件要求

项目	版本	下载地址
OpenMPI	4.0.1	https://www.open-mpi.org/software/ompi/v4.0/

操作系统要求

操作系统要求如表6-3所示。

表 6-3 操作系统要求

项目	版本	下载地址
CentOS	7.6	https://www.centos.org/download/

6.3 部署规划数据

本章节给出OpenMPI软件在安装部署过程中涉及到的相关软件安装规划路径的用途及详细说明。

表 6-4 部署规划数据

序号	软件安装规划路径	用途	说明
1	-	基础环境搭建中的各安装包安装路径。	参考《HPC解决方案 基础环境搭建指导书》中“安装规划数据”章节。
2	*/path/to/OPENMPI*	OPENMPI的安装规划路径。	这里的安装规划路径只是一个举例说明，建议部署在共享路径中。现网需要根据实际情况调整，后续章节凡是遇到安装路径的命令，都以现网实际规划的安装路径为准进行替换，不再单独说明。

6.4 配置安装环境

前提条件

使用SFTP工具将各安装包上传至服务器对应目录下。

配置流程

表 6-5 配置流程

序号	配置项	说明
1	基础环境搭建	参考《HPC解决方案 基础环境搭建指导书》中“单机场景环境搭建”章节。

6.5 部署 OpenMPI

6.5.1 下载安装包

操作步骤

步骤1 下载OpenMPI安装包“openmpi-4.0.1.tar.gz”。

下载地址：**https://download.open-mpi.org/release/open-mpi/v4.0/openmpi-4.0.1.tar.gz**。

步骤2 使用SFTP工具将OpenMPI安装包上传至服务器“*/path/to/OpenMPI*”目录下。

----结束

6.5.2 OpenMPI 编译安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令使用yum安装依赖包。

yum install numactl-devel-* systemd-devel-*

步骤3 执行以下命令加载编译器。

export PATH=*/path/to/GNU***/bin:$PATH**

export LD_LIBRARY_PATH= */path/to/GNU***/lib64:$LD_LIBRARY_PATH**

步骤4 执行以下命令解压OpenMPI安装包。

cd */path/to/OPENMPI*

tar -xvf openmpi-4.0.1.tar.gz

步骤5 执行以下命令进行配置。

cd openmpi-4.0.1

./configure --prefix=*/path/to/OPENMPI* **--enable-pretty-print-stacktrace --enable-orterun-prefix-by-default --with-knem=/opt/knem-1.1.3.90mlnx1/ --with-hcoll=/opt/mellanox/hcoll/ --with-cma --with-ucx --enable-mpi1-compatibility CC=gcc CXX=g++ FC=gfortran**

说明

- --with-ucx：使用系统自带的库“/usr/lib64/ucx”；
- --with-knem、--with-hcoll需要安装mellanox驱动，安装步骤请参考**《HPC解决方案 基础环境搭建指导书》**中“Infiniband网卡驱动安装”章节。

步骤6 执行以下命令进行安装编译。

make -j 16

make install

----结束

6.6 验证 OpenMPI

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令加载环境变量。

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**

步骤3 执行以下命令验证OpenMPI是否安装成功。

mpirun --version

显示如下表示安装成功。

```
mpirun (Open MPI) 4.0.1
Report bugs to http://www.open-mpi.org/community/help/
```

----结束

7 Slurm 18.08.7 安装指南

7.1 介绍

简要介绍

Slurm是一个开源，高度可扩展的集群管理工具和作业调度系统，用于各种规模的Linux集群。主要提供如下集中关键的特性。

资源分配

分配独占或者非独占的资源给用户，可以控制分配的时长，供用户运行作业。

作业管理框架

提供一个框架，可以帮助用户控制并行作业在所分配资源上的启动、运行和监控。

队列

提交的作业资源需求超出了可用资源，将作业放入队列。

不同的作业调度策略

提供资源预留，公平分享，回填等高级作业调度策略供使用。

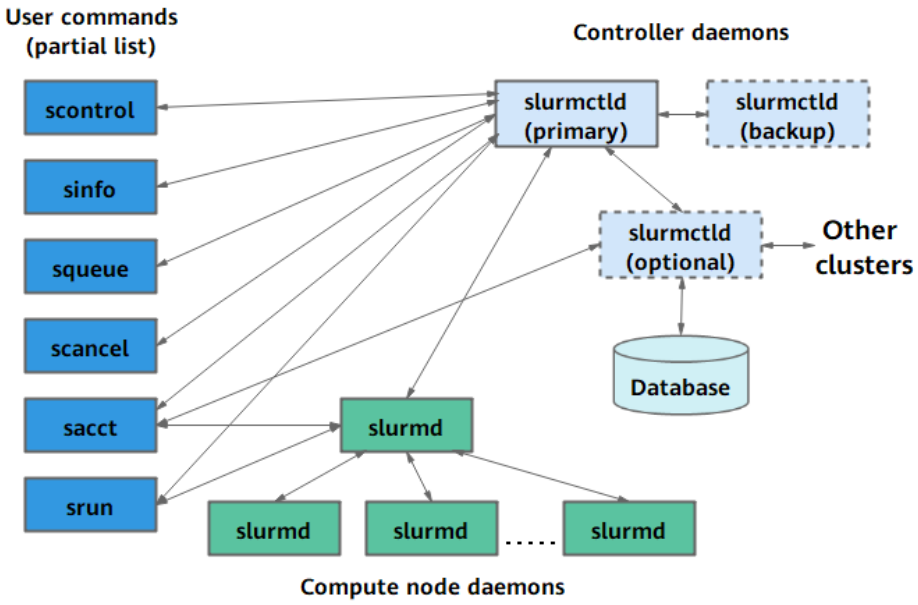
其他工具

提供作业信息统计，作业状态诊断等工具。

建议的版本

建议使用版本为“Slurm 18.08.7”。

软件架构



Slurm有一个集中式的管理进程，“slurmctld”，来监控资源和作业；

每个计算节点有一个“slurmd”守护进程，用来等待接受作业、执行作业、返回结果、再等待下一个作业；

“slurmdbd”是可选的，用于在一个数据库中记录多个slurm管理集群的作业统计信息；

详细信息参考如下链接：

https://slurm.schedmd.com/overview.html

7.2 环境信息

硬件要求

硬件要求如表7-1所示。

表 7-1 硬件要求

项目	说明
CPU	Kunpeng 920

软件要求

软件要求如表7-2所示。

表 7-2 软件要求

项目	版本	下载地址
Slurm	18.08.7	https://www.schedmd.com/downloads.php
munge	0.5.13	https://github.com/dun/munge/releases/tag/munge-0.5.13

操作系统要求

操作系统要求如表7-3所示。

表 7-3 操作系统要求

项目	版本	下载地址
CentOS	7.6	https://www.centos.org/download/

集群信息

集群信息如表7-4所示。

表 7-4 集群信息

节点名	IP	作用
master	192.168.40.11	管理节点，运行slurmctld
testnode1	192.168.40.111	计算节点，运行slurmd
testnode2	192.168.40.112	计算节点，运行slurmd

7.3 部署规划数据

本章节给出Slurm软件在安装部署过程中涉及到的相关软件安装规划路径的用途及详细说明。

表 7-5 部署规划数据

序号	软件安装规划路径	用途	说明
1	*/path/to/SLURM*	Slurm的安装规划路径。	这里的安装规划路径只是一个举例说明，建议部署在共享路径中。现网需要根据实际情况调整，后续章节凡是遇到安装路径的命令，都以现网实际规划的安装路径为准进行替换，不再单独说明。
2	*/path/to/MUNGE*	munage的安装规划路径。	

7.4 Slurm 的 rpm 包生成

7.4.1 下载安装包

步骤1 下载Munge安装包“munge-0.5.13.tar.xz”。

下载地址：**https://github.com/dun/munge/releases/tag/munge-0.5.13**

步骤2 下载Slurm安装包“slurm-18.08.7.tar.bz2”。

下载地址：**https://download.schedmd.com/slurm/slurm-18.08.7.tar.bz2**

步骤3 使用SFTP工具：

- 将Munge安装包上传至服务器“*/path/to/MUNGE*”目录。
- 将Slurm安装包上传至服务器“*/path/to/SLURM*”目录。

----结束

7.4.2 依赖库安装

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令安装依赖库。

yum install -y rpm-build rpmdevtools bzip2-devel openssl-devel zlib-devel readline-devel pam-devel perl-DBI perl-ExtUtils-MakeMaker mariadb*

----结束

7.4.3 编译 Munge

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令构建munge的rpm包。

cd */path/to/MUNGE*

rpmbuild -tb --clean munge-0.5.13.tar.xz

步骤3 执行以下命令检查是否成功生成rpm包。

ls /root/rpmbuild/RPMS/aarch64/ | grep munge

```
munge-0.5.13-1.el7.aarch64.rpm
munge-debuginfo-0.5.13-1.el7.aarch64.rpm
munge-devel-0.5.13-1.el7.aarch64.rpm
munge-libs-0.5.13-1.el7.aarch64.rpm
```

步骤4 在*“/path/to/MUNGE”*目录下创建一个“mungerpm”目录，并将“/root/rpmbuild/RPMS/aarch64/”目录下munge的rpm包拷贝到“/path/to/MUNGE/mungerpm”目录。

mkdir -p */path/to/MUNGE/***mungerpm**

cp /root/rpmbuild/RPMS/aarch64/munge* */path/to/MUNGE/***mungerpm –f**

----结束

7.4.4 编译 Slurm

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令进入Munge目录。

cd /home/mungerpm

步骤3 执行以下命令安装Munge。

yum install -y munge-*

步骤4 执行以下命令进入Slurm目录。

cd */path/to/SLURM*

步骤5 执行以下命令构建Slurm的rpm包。

rpmbuild -ta --clean slurm-18.08.7.tar.bz2

步骤6 执行以下命令检查是否成功生成rpm包。

ls /root/rpmbuild/RPMS/aarch64/ | grep slurm

```
slurm-18.08.7-1.el7.aarch64.rpm
slurm-contribs-18.08.7-1.el7.aarch64.rpm
slurm-devel-18.08.7-1.el7.aarch64.rpm
slurm-example-configs-18.08.7-1.el7.aarch64.rpm
slurm-libpmi-18.08.7-1.el7.aarch64.rpm
slurm-openlava-18.08.7-1.el7.aarch64.rpm
slurm-pam_slurm-18.08.7-1.el7.aarch64.rpm
slurm-perlapi-18.08.7-1.el7.aarch64.rpm
slurm-slurmctld-18.08.7-1.el7.aarch64.rpm
slurm-slurmd-18.08.7-1.el7.aarch64.rpm
slurm-slurmdbd-18.08.7-1.el7.aarch64.rpm
slurm-torque-18.08.7-1.el7.aarch64.rpm
```

步骤7 在*“/path/to/SLURM”*目录下创建一个“slurmrpm”目录，并将“/root/rpmbuild/RPMS/aarch64/”目录下slurm的rpm包拷贝到“/path/to/SLURM/slurmrpm”目录。

mkdir -p */path/to/SLURM/***slurmrpm**

cp /root/rpmbuild/RPMS/aarch64/slurm* */path/to/SLURM/***slurmrpm -f**

----结束

7.5 Slurm 的安装配置

7.5.1 安装 Munge

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令在testnode1和testnode2节点上挂载master节点的“/home”目录。

mount master:/home /home

步骤3 执行以下命令在testnode1和testnode2节点上安装munge相关包。

cd /home/mungerpm

yum install -y munge*

步骤4 执行以下命令修改master、testnode1和testnode2节点相应munge目录的权限。

chmod -Rf 700 /etc/munge

chmod -Rf 711 /var/lib/munge

chmod -Rf 700 /var/log/munge

chmod -Rf 0755 /var/run/munge

步骤5 执行以下命令在master节点启动ntpd服务。

yum install -y ntp

systemctl start ntpd

步骤6 执行以下命令在testnode1和tesnode2节点上，将系统时间同步到master节点。

ntpdate master

步骤7 在master节点上，将master节点的“/etc/munge/munge.key”拷贝到testnode1和testnode2节点。

scp /etc/munge/munge.key testnode1:/etc/munge/

scp /etc/munge/munge.key testnode2:/etc/munge/

步骤8 在testnode1和testnode2节点上，更改“/etc/munge/munge.key”文件的权限。

chown munge.munge /etc/munge/munge.key

步骤9 在master、testnode1和testnode2节点上启动munge。

systemctl start munge

systemctl enable munge

----结束

7.5.2 安装 Slurm

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令在master、testnode1和testnode2节点上安装slurm相关包。

cd /home/slurmrpm

yum install -y slurm*

步骤3 检查所有节点系统中是否已经创建slurm用户。

- 如果已经创建，则执行以下命令进行查看：

  grep "slurm" /etc/group

  ```
  slurm:x:202:
  ```

- 如果没有创建，则执行以下命令在master、testnode1和testnode2节点上创建slurm用户。

  groupadd -g 202 slurm

  useradd -u 202 -g 202 slurm

步骤4 执行以下命令在master、testnode1和testnode2节点下创建“/var/spool/slurm/ssl”目录、“/var/spool/slurm/d”目录和“/var/log/slurm”目录。

mkdir -p /var/spool/slurm/ssl

mkdir -p /var/spool/slurm/d

mkdir -p /var/log/slurm

步骤5 执行以下命令在master、testnode1和testnode2节点上设置相应目录权限。

chown -R slurm.slurm /var/spool/slurm

步骤6 执行以下命令修改matser节点上的“/etc/slurm/slurm.conf”文件。

1. **vi /etc/slurm/slurm.conf**
2. 按“i”进入编辑模式，添加如下内容。

```
ControlMachine=master
ControlAddr=192.168.40.11
MpiDefault=none
ProctrackType=proctrack/pgid
ReturnToService=1
SlurmctldPidFile=/var/run/slurmctld.pid
SlurmdPidFile=/var/run/slurmd.pid
SlurmdSpoolDir=/var/spool/slurm/d
SlurmUser=slurm
#SlurmdUser=root
StateSaveLocation=/var/spool/slurm/ssl
SwitchType=switch/none
TaskPlugin=task/none
FastSchedule=1
SchedulerType=sched/backfill
SelectType=select/linear
AccountingStorageType=accounting_storage/none
ClusterName=cluster
JobAcctGatherType=jobacct_gather/none
SlurmctldDebug=3
SlurmctldLogFile=/var/log/slurm/slurmctld.log
SlurmdDebug=3
```

```
SlurmdLogFile=/var/log/slurm/slurmd.log

NodeName=testnode1 CPUs=96 Sockets=4 CoresPerSocket=24 State=UNKNOWN
NodeName=testnode2 CPUs=40 Sockets=4 CoresPerSocket=10 State=UNKNOWN

PartitionName=ARM Nodes=testnode1 Default=YES MaxTime=INFINITE State=UP
PartitionName=X86 Nodes=testnode1 Default=YES MaxTime=INFINITE State=UP
```

3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

步骤7 执行以下命令在master节点上，将master节点的“/etc/slurm/slurm.conf”拷贝到testnode1和testnode2节点。

scp /etc/slurm/slurm.conf testnode1:/etc/slurm

scp /etc/slurm/slurm.conf testnode2:/etc/slurm

步骤8 执行以下命令在master节点启动“slurmctld”服务。

systemctl start slurmctld

systemctl enable slurmctld

步骤9 执行以下命令在testnode1和testnode2节点启动“slurmd”服务。

systemctl start slurmd

systemctl enable slurmd

----结束

7.6 Slurm 的使用

7.6.1 常用命令

常用命令如所示。

表 7-6 常用命令

命令	说明
sinfo	查看节点与分区状态
squeue	查看任务队列状态
scontrol show nodes	详细显示节点信息
scontrol show jobs	详细显示任务信息
srun	执行作业
salloc	分配资源
sbatch	提交批处理作业
scancel	取消作业

7.6.2 批处理脚本作业

7.6.2.1 脚本编写

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令编写脚本。

1. **vi test.slurm**
2. 按“i”进入编辑模式，添加如下内容。

```
#!/bin/bash
#SBATCH -J slurmtest
#SBATCH -o %J.out
#SBATCH -e %J.err
#SBATCH -N 1
#SBATCH --exclusive
#SBATCH -p X86

for x in {0..10}
do
echo "This is a slurmtest for `arch`!"
sleep 5
done
```

3. 按“Esc”键，输入**:wq!**，按“Enter”保存并退出编辑。

----结束

7.6.2.2 作业提交

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令提交作业。

sbatch test.slurm

----结束

7.6.2.3 作业取消

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令查看作业ID号。

squeue

步骤3 执行以下命令取消作业。

scancel +*ID号*

示例：**scancel 123**

----结束

7.6.2.4 常见参数解析

常见参数如表7-7所示。

表 7-7 常见参数

参数	含义
-J	指定作业名
-o	指定标准输出的保存文件
-e	指定标准错误输出的保存文件
--nodes / -N	指定要求的节点数
--ntasks / -n	指定总进程数
--exclude	指定哪些节点不允许使用
--nodelist	指定使用的节点列表
--time	指定该job允许运行的最长时间
--exclusive	指定已分配的节点独占使用
--partition/-P	指定某些partition内的节点才能分配给该job
-d singleton	同名同时只运行一个文件

7.6.3 Slurm 节点状态

操作步骤

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令在master节点上使用sinfo命令查看节点的状态。

sinfo

```
PARTITION    AVAIL  TIMELIMIT  NODES  STATE NODELIST
ARM          up   infinite1 idletestnode1
X86*  up     infinite1 idletestnode2
```

表 7-8 参数说明

参数	含义
unk	未知
idle	空闲
alloc	已分配
down	故障

----结束

7.6.4 更多信息

Slurm更多配置信息及参数解析，请查阅slurm官网：**https://slurm.schedmd.com/documentation.html**。

7.7 故障排除

7.7.1 计算节点状态异常问题

问题描述：

计算节点处于“down”状态，且无法自行恢复。

处理步骤：

步骤1 使用PuTTY工具，以root用户登录服务器。

步骤2 执行以下命令手动恢复节点状态。

scontrol update nodename=testnode1 state=resume

步骤3 执行以下命令检查节点状态。

sinfo

```
PARTITION    AVAIL  TIMELIMIT  NODES  STATE  NODELIST
ARM          up     infinite   1      idle   testnode1
X86*         up     infinite   1      idle   testnode2
```

----结束

8 修订记录

发布日期	修订记录
2020-03-20	第一次正式发布。

