HPC Cloud Solution

User Manual

Issue 01

Date 2025-09-29

HUAWEI TECHNOLOGIES CO., LTD.

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 Overview

1.1 About HPC

Introduction

High-performance computing (HPC) is a computer cluster system that connects multiple computer systems using various interconnection technologies. It relies on the integrated computing capability of all the connected systems to perform large-scale computing tasks. For this reason, HPC is also often referred to as an HPC cluster.

HPC Service Characteristics

In industries including scientific research, weather forecast, simulation test, bio-pharmaceuticals, gene sequencing, and image processing, high-performance clusters are required to resolve large-scale computing issues. The management node assigns computing tasks to different computing nodes.

In different service scenarios, the data volume and correlations between computing tasks vary, posing varied requirements for computing capabilities, storage efficiency, network bandwidth, and delay.

HPC Application Scenarios

HPC provides ultra-high floating-point computing capabilities, applicable in massive data processing and computing-intensive scenarios such as scientific research, weather forecasting, computing simulation, military research, CAD/CAE, biopharmaceuticals, DNA sequencing, and image processing. It shortens the time it takes to perform tasks and enhances computing precision.

1.2 HPC Management and Scheduling Plug-in

Product Overview

The HPC management and scheduling plug-in is an end-to-end one-stop Huawei Cloud cluster resource usage and management platform that is developed based on Slurm. It provides one-click cluster delivery on a visualized interaction interface. It integrates SFS Turbo file systems to provide high-performance shared storage. You can perform operations on cluster users, compute resources, and service jobs on the UI. The plug-in supports quick modeling and computing in scenarios such as structural mechanics, fluid analysis, thermal simulation, and gene sequencing.

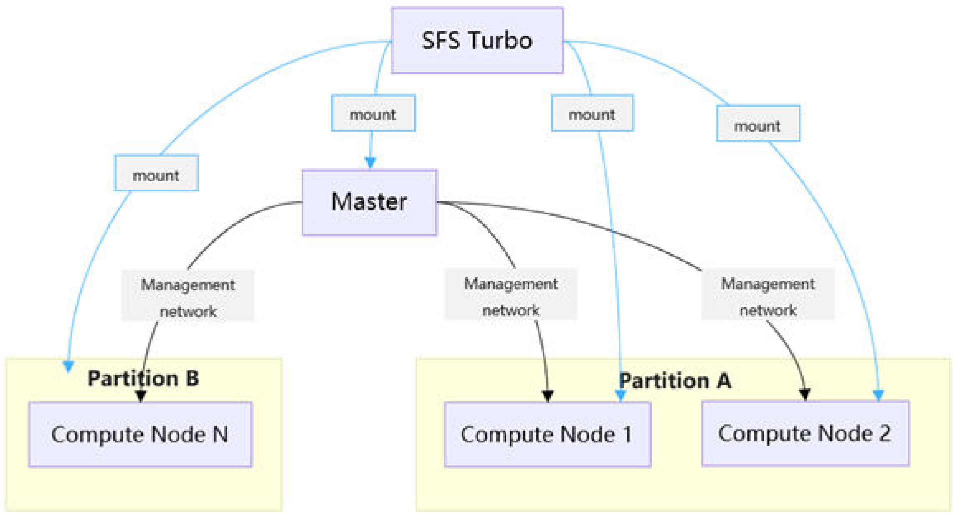
Core Functions

Function	Description
Partition Management	Divides logical resource pools and isolates resources of different teams or projects.
Cluster Management	Creates, destroys, and manages compute resources and monitors cluster metrics.
Topology Management	Defines the physical topology structure (such as racks and switches) of a cluster and optimizes job scheduling policies.
Job Management	Submits tasks based on user requirements and queries task logs, job status, completion time, and scheduled nodes.
Job Templates	Sets standard job configurations and submits tasks in one click.
Elastic Resource Supply	Configures at least one scaling policy for each partition to automatically scale in or out compute nodes based on the policy.
Elastic Job Scheduling	Schedules jobs based on policies, such as by priority, first in, first out (FIFO), and backfill scheduling.
Quota Management	Restricts resource usages of users or groups by QoS, accounts, and partitions to ensure fair access and prioritized use.
Data Management	Mounts SFS Turbo file systems to provide high-performance shared storage. Files smaller than 1 GB can be uploaded and downloaded on the cockpit UI.
Tag Management	Adds tags to nodes for fine-grained management of resources in the same partition.
Auditing Management	Logs user operations and resource usages.
Cluster O&M	Allows you to view node processes, system configurations, environment variables, and downloaded logs.
User Management	Has a built-in administrator account that can be used to create and delete common users. These users are assigned different roles to access the cluster UIs.

System Architecture and Deployment Requirements

Architecture Topology

- Management and control nodes
  - Master node: has 16 vCPUs, 32-GB memory, and a 300-GB disk and is responsible for cluster scheduling, user management, and audit log storage
  - SFS Turbo: provides a shared file system. The mount path is **/mnt/sfs_turbo_1**.
- Compute nodes: Pay-per-use or yearly/monthly compute nodes are created on the cockpit UI or using elastic policies.



Deployment Requirements

Component	Configuration Requirements
Master node	16 vCPUs, 32-GB memory, a 300-GB SSD disk, and associated with an EIP
SFS Turbo	On-demand capacity expansion by at least 1 TB and bandwidth of at least 1 Gbit/s
Compute nodes	You can create compute nodes on the cockpit UI and select specifications as required.

Deployment

For details, see **Deployment**.

Other Advanced Features

- **Capacity Reservation**
- **Bursting from on-premises**

- **Bin packing**

1.3 HPC and Public Cloud

Advantages of Deploying HPC in the Public Cloud

Traditional HPC encounters the following issues:

- Investment cost is high, expansion is complex, and reusing existing investment is difficult.
- Applications are complicated, resource prediction is difficult, flexibility is poor, and efficiency needs to be improved.
- Decision making is slow, causing enterprises to lose the market share and opportunities to develop research results.
- The application calculation amount is increasing rapidly, posing a higher requirement for performance.

Using HPC in public cloud helps make full use of the following advantages of cloud services:

- Reduced TCO

  Allows customers to pay as they use, reducing costs and lowering the threshold for small and medium-sized enterprises to use HPC.
- Improved efficiency

  Supports on-demand provisioning as well as quick deployment and capacity expansion, shortening the time to market (TTM) and scientific research period for products.
- Flexible use
  - Presets the MPI library, compilation library, and optimized configuration in the image template to accelerate environment deployment.
  - Allows enterprise branches and R&D organizations in different geographical locations to coordinate with each other easily, thereby improving efficiency.
  - Allows customers to use the cross-region capabilities of the public cloud to share computing resources and massive data, and analyze big data in the cloud.
- Optimized performance
  - Performance is 30% higher than common ECSs.
  - According to test reports, with virtualization optimization measures such as SR-IOV and PCI passthrough adopted in large-scale cloudification, HPC performance is not deteriorated significantly.

Relationship Between HPC and Cloud Services

Table 1-1 Cloud services

Cloud Service	Function
ECS	Used for creating high-performance ECSs on the public cloud platform.
VPC	All ECSs in HPC scenarios are located in the same VPC. They are isolated for network security using VPC subnets and network groups.
Image Management Service (IMS)	● Image files are required for creating high-performance ECSs. ● High-performance ECSs are used to create private images.
Elastic Volume Service (EVS)	All ECSs used in HPC scenarios have EVS disks attached.
Bare Metal Server (BMS)	A BMS is a physical server dedicated for individual tenants. It provides remarkable computing performance and stability for running key applications.
Object Storage Service (OBS)	The Object Storage Service (OBS) is an object-based mass cloud storage service. It provides massive, low-cost, highly reliable, and secure data storage capabilities.
Scalable File Service (SFS)	SFS provides ECSs with hosted shared file storage that complies with the standard file protocol (NFS) and that can scale up to the PB level to support massive data and high-bandwidth applications.
Data Express Service (DES)	DES is a massive data transmission service that uses the physical storage medium (USB or eSATA interface) to transmit a large amount of data to Huawei public cloud. It resolves the high cost and long time consumed by massive data transmission over the Internet.
Direct Connect	Direct Connect is a service that allows you to establish a dedicated network connection from your data center to the Huawei public cloud. It enables you to take full advantage of the strengths of public cloud services while continue to use existing IT facilities, and establish a scalable hybrid cloud computing environment.

Cloud Service	Function
Cloud Eye (CES)	CES provides a comprehensive monitoring platform for resources such as the ECS and bandwidth. It enables you to monitor alarms, receive notifications, and view reports and visuals in real time to obtain the status of service resources.
Workspace	Workspace provides virtual desktops and applications and allows you to use cloud desktops anywhere and anytime. It provides professional office applications and a more simplified and secure IT office system with higher service efficiency and lower maintenance cost.

1.4 IPoIB Functions

IPoIB

Internet Protocol over InfiniBand (IPoIB) allows connection and data transmission through physical IB networks (IB card, cable, and switch on the server) over the IP protocol.

It provides a RDMA-based IP network simulation layer and allows you to run applications on the InfiniBand network without making any modification. However, IPoIB is inferior to RDMA in performance. Most applications use RDMA that provides high bandwidth and low latency, and some key applications use IPoIB.

NOTE

Applications can run on the InfiniBand network over the IP protocol without any modification required.

Communication Modes

Two modes datagram and connected can be configured for IPoIB devices. The former provides unreliable and connectionless links, and the latter provides reliable links with connections.

- In the datagram mode, the queue pair does not allow the packet size to exceed the MTU value of the IB link layer. The IPoIB header contains four bytes, and the IPoIB MTU value is less than that of the IB link layer.
- In the connected mode, the queue pair allows packets larger than those of the IB link layer. In theory, the packet can contain a maximum of 65535 bytes. The connected mode has better performance, but consumes more memory. Most systems give higher priority to performance, and the connected mode is configured for the IB network port.

The NIC driver of the current version does not support the connected mode.

NOTE

Due to NIC driver performance issues, the connected mode is disabled. Therefore, the NIC driver in the HPC solution does not support the connected mode.

IP Address Assignment Methods

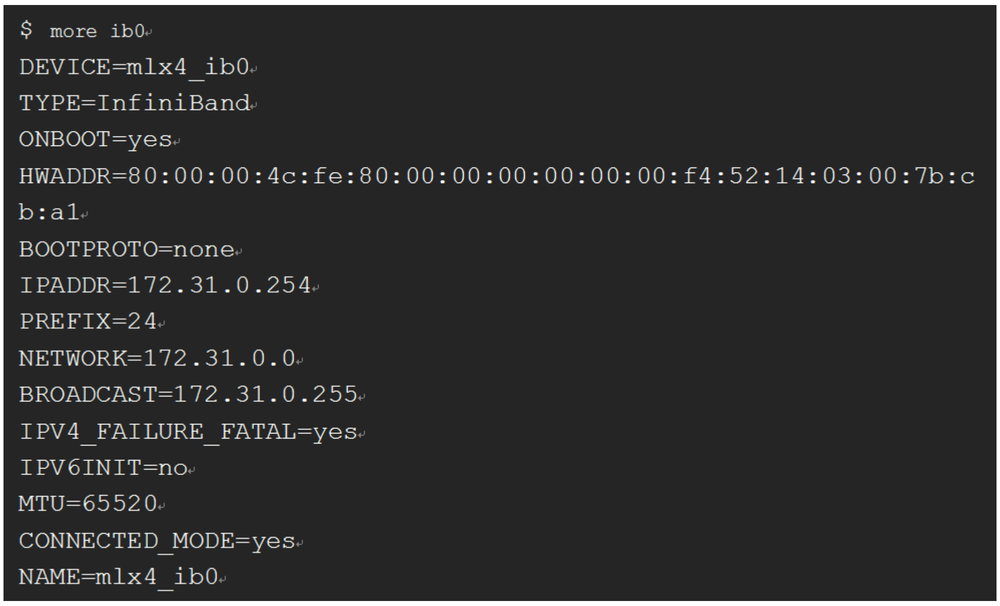
Two methods are available, static configuration and DHCP.

- Static configuration

  IPoIB devices have a hardware address containing 20 bytes. The first four bytes are the queue pair number, the middle eight bytes is the subnet prefix, and the last eight bytes is the GUID.

  You can query the hardware address of an IPoIB device only by running the **ip** command. If you run the **ifconfig** command, you cannot obtain a complete address. The following figure shows an example of a static IP address.

  Figure 1-1 Static IP address



```
$ more ib0
DEVICE=mlx4_ib0
TYPE=InfiniBand
ONBOOT=yes
HWADDR=80:00:00:4c:fe:80:00:00:00:00:00:00:f4:52:14:03:00:7b:c
b:a1
BOOTPROTO=none
IPADDR=172.31.0.254
PREFIX=24
NETWORK=172.31.0.0
BROADCAST=172.31.0.255
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
MTU=65520
CONNECTED_MODE=yes
NAME=mlx4_ib0
```

- DHCP

  A standard DHCP frame contains fields, such as hardware type (**htype**), MAC address length (**hlen**), and MAC address (**chaddr**). The MAC address is not long enough to contain the IPoIB hardware address. Therefore, field **client-identifier** is defined to identify DHCP sessions on the client. The **client-identifier** field is used to associate the IP address with the client. DHCP Server uses the field to distinguish IP addresses assigned to different clients.

  The HPC solution uses the DHCP method to ensure that IP addresses are automatically assigned and configured.

Constraints

- One IB NIC can be managed.
- Similar to constraints of the InfiniBand NICs of BMSs, H2, HL1, and HI3 ECSs, ECSs using IPoIB cannot be migrated.

- Similar to constraints of the InfiniBand NICs of H2, HL1, and HI3 ECSs, InfiniBand networks do not support security groups, QoS, and layer-3 and higher-layer network functions.
- ECSs using IPoIB do not support ARP anti-spoofing and DHCP anti-spoofing due to NIC driver limitations.

1.5 Quota Adjustment

What Is Quota?

Quotas can limit the number or amount of resources available to users, such as the maximum number of ECS or EVS disks that can be created.

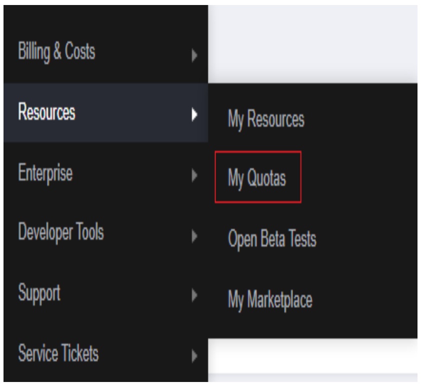
If the existing resource quota cannot meet your service requirements, you can apply for a higher quota.

How Do I View My Quotas?

1. Log in to the **management console**.
2. Click in the upper left corner and select the desired region and project.
3. In the upper right corner of the page, choose **Resources** > **My Quotas**.

   The **Quotas** page is displayed.

   Figure 1-2 My Quotas

4. View the used and total quota of each type of resources on the displayed page.

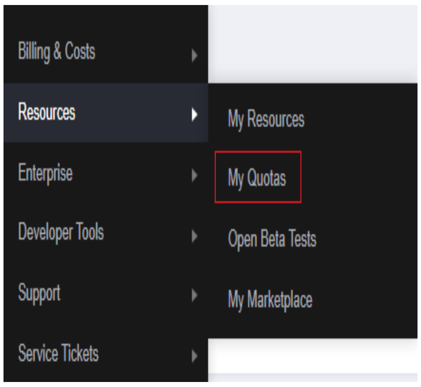
   If a quota cannot meet service requirements, apply for a higher quota.

How Do I Apply for a Higher Quota?

1. Log in to the **management console**.
2. In the upper right corner of the page, choose **Resources** > **My Quotas**.

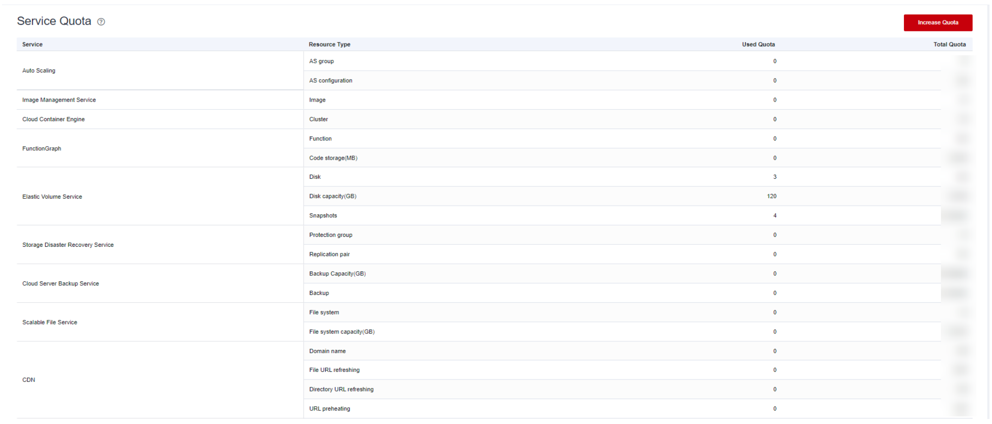
   The **Quotas** page is displayed.

Figure 1-3 My Quotas



3. Click **Increase Quota** in the upper right corner of the page.

Figure 1-4 Increasing quota



Service Quota

Increase Quota

Service	Resource Type	Used Quota	Total Quota
Auto Scaling	AS group	0	[redacted]
	AS configuration	0	[redacted]
Image Management Service	Image	0	[redacted]
Cloud Container Engine	Cluster	0	[redacted]
FunctionGraph	Function	0	[redacted]
	Code storage(MB)	0	[redacted]
Elastic Volume Service	Disk	3	[redacted]
	Disk capacity(GB)	120	[redacted]
	Snapshots	4	[redacted]
Storage Disaster Recovery Service	Protection group	0	[redacted]
	Replication pair	0	[redacted]
Cloud Server Backup Service	Backup Capacity(GB)	0	[redacted]
	Backup	0	[redacted]
Scalable File Service	File system	0	[redacted]
	File system capacity(GB)	0	[redacted]
CDN	Domain name	0	[redacted]
	File URL refreshing	0	[redacted]
	Directory URL refreshing	0	[redacted]
	URL preheating	0	[redacted]

4. On the **Create Service Ticket** page, configure parameters as required.

   In the **Problem Description** area, fill in the content and reason for adjustment.

5. After all necessary parameters are configured, select **I have read and agree to the Ticket Service Protocol and Privacy Statement** and click **Submit**.

2 Typical Applications in the ECS Scenario

2.1 Creating an ECS That Supports InfiniBand NICs

Scenarios

You can obtain scalable ECSs on the public cloud platform within minutes based on requirements. This section describes how to create an ECS that supports InfiniBand NICs both on the management console and by calling HTTPS-based APIs.

Through the Management Console

1. Log in to the management console.
2. Under **Computing**, click **Elastic Cloud Server**.

   The **Elastic Cloud Server** page is displayed.
3. Click **Create ECS**.
4. Configure basic information about the ECS to be created. For details, see **Table 2-1**.

Table 2-1 Parameter description

Parameter	Description	Example Value
Region	If the region is incorrect, click 📍 in the upper left corner of the page for correction.	AP-Singapore

Parameter	Description	Example Value
AZ	An AZ is a physical location where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network. ● To enhance application availability, create ECSs in difference AZs. ● To shorten network latency, create ECSs in the same AZ.	az-01
Specifications	Select the H2 or HI3 ECS.	h2.4xlarge.8
DeH	Physical host resources dedicated for a specified user. This parameter is not required. HPC involves only one ECS on a host, and no DeH is required.	N/A

Parameter	Description	Example Value
Image	● Public image A public image is a standard, widely used image. It contains an OS and preinstalled public applications and is available to all users. You can configure the applications or software in the public image as needed. To select a public image, set **Image** to **Public image** and select a desired one from the drop-down lists. ● Private image A private image is an image available only to the user who created it. It contains an OS, preinstalled public applications, and the user's private applications. Using a private image to create ECSs removes the need to configure multiple ECSs repeatedly. To select a private image, set **Image** to **Private image** and select a desired one from the drop-down list. You can also select an encrypted image. For details, see *Image Management Service User Guide*. ● Shared image A shared image is a private image shared by another public cloud user. To select a shared image, set **Image** to **Shared image** and select a desired one from the drop-down list. ● Marketplace image A Marketplace image is a third-party image that has the OS, application environment, and software pre-installed. You can use the images to deploy websites and application development environments with a few clicks. No additional configuration operation is required. To select a Marketplace image, set **Image** to **Marketplace image**, click **Select Image** following the **Image** text box, and select a desired one in the displayed dialog box.	Public image

Parameter	Description	Example Value
License Type	Specifies a license type for using an OS or software on the public cloud platform. This parameter is optional. If the image you selected is free of charge, this parameter is unavailable. If the image you selected is charged, such as a SUSE, Oracle Linux, or Red Hat image, this parameter is available. ● Use the system license Allows you to use the license provided by the public cloud platform. Obtaining the authorization of such a license is charged. ● Bring your own license (BYOL) Allows you to use your existing OS license. In such a case, you do not need to apply for a license again.	Bring your own license (BYOL)

Parameter	Description	Example Value
Disk	Also called the EVS disk, which can be a system disk or data disk. ● System Disk If the image based on which an ECS is created is not encrypted, the system disk of the ECS is not encrypted. In addition, **Unencrypted** is displayed for the system disk on the page. If the image based on which an ECS is created is encrypted, the system disk of the ECS is automatically encrypted. For details, see section **(Optional) Encryption-related parameters**. ● Data Disk You can create multiple data disks for an ECS and configure sharing and encryption functions as well as device type for each data disk. – **SCSI**: indicates that the device type of the data disk is SCSI. SCSI EVS disks support transparent SCSI command transmission and allow the server OS to directly access the underlying storage media. In addition to supporting simple SCSI I/O commands, SCSI EVS disks support advanced SCSI commands. **NOTE** If **SCSI** is not selected, VBD EVS disks are created by default, which support only simple SCSI read-write commands. – **Share**: indicates that the EVS disk is shared. Such an EVS disk can be attached to multiple ECSs. – **Encryption**: indicates that the data disk is encrypted. For details, see section **(Optional) Encryption-related parameters**. ● (Optional) Encryption-related parameters To enable encryption, click **Create Xrole** to grant KMS access rights to EVS. If you have rights granting permission, grant the KMS access rights to EVS. If you do not have the permission, contact the user having the security administrator rights to grant the KMS access rights.	System disk: ultra-high I/O, 40 GB

Parameter	Description	Example Value
	– **Encrypted**: indicates that the EVS disk has been encrypted. – **Create Xrole**: grants KMS access rights to EVS to obtain KMS keys. After the rights are granted, follow-up operations do not require rights granting again. – **KMS Key Name**: specifies the name of the key used by the encrypted EVS disk. By default, the name is **evs/default**. – **Xrole Name: EVSAccessKMS**: indicates that rights have been granted to EVS to obtain KMS keys for encrypting or decrypting EVS disks. – **KMS Key ID**: specifies the ID of the key used by the encrypted data disk. For details about EVS disk types, device types, shared EVS disks, and encryption, see *Elastic Volume Service User Guide*.	

5. Set network parameters, including **VPC**, **Security Group**, **NIC**, and **EIP**.

   When you use VPC for the first time, the system automatically creates a VPC for you, including the security group and NIC.

Table 2-2 Parameter description

Parameter	Description	Example Value
VPC	Provides a network, including subnet and security group, for an ECS. You can select an existing VPC, or click **View VPC** and create a desired one. **NOTE** ECSs in an HPC cluster must belong to the same VPC and subnet.	N/A

Parameter	Description	Example Value
Security Group	Controls ECS access within a security group or between security groups by defining access rules. You can define different access control rules for a security group, and these rules take effect for all ECSs added to this security group. When creating an ECS, you can select multiple security groups (no more than five is recommended). In such a case, the access rules of all the selected security groups apply on the ECS. **NOTE** Before initializing an ECS, ensure that security group rules in the outbound direction meet the following requirements: ● **Protocol**: **TCP** ● **Port Range**: **80** ● **Remote End**: **169.254.0.0/16** If you use the default security group rule in the outbound direction, the preceding requirements are met, and the ECS can be initialized. The default security group rule in the outbound direction is as follows: ● **Protocol**: **ANY** ● **Port Range**: **ANY** ● **Remote End**: **0.0.0.0/16**	N/A
NIC	Includes primary and extension NICs. You can add multiple expansion NICs to an ECS and specify IP addresses for them (including primary NICs).	N/A
EIP	A static public IP address bound to an ECS in a VPC. Using the EIP, the ECS provides services externally. The following options are provided: ● Auto assign: The system automatically assigns an EIP for the ECS. The EIP provides exclusive bandwidth that is configurable. ● : An existing EIP is assigned for the ECS. When using an existing EIP, you cannot create ECSs in batches.	Auto assign

6. Set **Login Mode**.

   Key pair is recommended because it features higher security than **Password**. If you select **Password**, ensure that the password meets complexity requirements listed in **Table 2-3** to prevent malicious attacks.

– Key pair

  A key pair is used for ECS login authentication. You can select an existing key pair, or click **View Key Pair** and create a desired one.

  NOTE

  If you use an existing key pair, make sure that you have saved the key file locally. Otherwise, logging in to the ECS will fail.

– Password

  If you choose the initial password for authentication in an ECS, you can log in to an ECS using the username and its initial password.

  The initial password of user **root** is used for authentication in Linux, while that of user **Administrator** is used for authentication in Windows.

Table 2-3 Password complexity requirements

Parameter	Rules
Password	● Consists of 8 to 26 characters. ● Must contain at least three of the following character types: – Uppercase letters – Lowercase letters – Digits – Special characters for Windows ECSs: !@$%^-_=+[{()}]:,./?~#* – Special characters for Linux ECSs: !@$%^-_=+[{}]:,./?~#* ● Cannot contain the username or the username spelled backwards. ● Cannot contain more than two consecutive characters in the same sequence as they appear in the username. (This requirement applies only to Windows ECSs.)

NOTE

The system does not automatically change the password for logging in to an ECS on a regular basis. It is recommended that you change your password regularly for security.

7. Configure **Advanced Settings**.

   To use functions listed in **Advanced Settings**, click **Configure now**. Otherwise, click **Do not configure**.

   – File Injection

     Enables the system to automatically inject a script file or other files into a specified directory on an ECS when you create the ECS. This configuration is optional. After the file injection function is enabled, the system automatically injects files into a specified directory when creating an ECS.

- User Data Injection

  Enables the ECS to automatically inject user data when the ECS starts for the first time. This configuration is optional. After this function is enabled, the ECS automatically injects the user data upon its first startup.
- ECS Group

  An ECS group applies the anti-affinity policy to the ECSs in it so that the ECSs can be distributed on different hosts.

  NOTE

  If you use a shared EVS disk of the SCSI type as the data disk, you are suggested to configure an ECS group for the ECS to be created to support SCSI-locking commands.
- Tag

  Tags an ECS, facilitating ECS identification and management.

  This configuration is optional.

8. Set **ECS Name**.

   The name can be customized but can contain only letters, digits, underscores (_), hyphens (-), and periods (.).

   If you want to create multiple ECSs at a time, the system automatically sequences these ECSs.
9. Configure the number of ECSs to be created.

   After the configuration, click **Price Calculator** to view the ECS configuration fee.
10. Click **Next**.
11. On the page for you to confirm ECS configurations, view details about the ECS.

    After confirming ECS configurations, click **Submit**.

    After an ECS is created, you can view information about it on the **Elastic Cloud Server** page.
12. (Optional) If you create the ECS with a data disk added, initialize the disk after the ECS is created.

    For details, see section "Initializing an EVS Data Disk" in *Elastic Volume Server User Guide*.

Through APIs

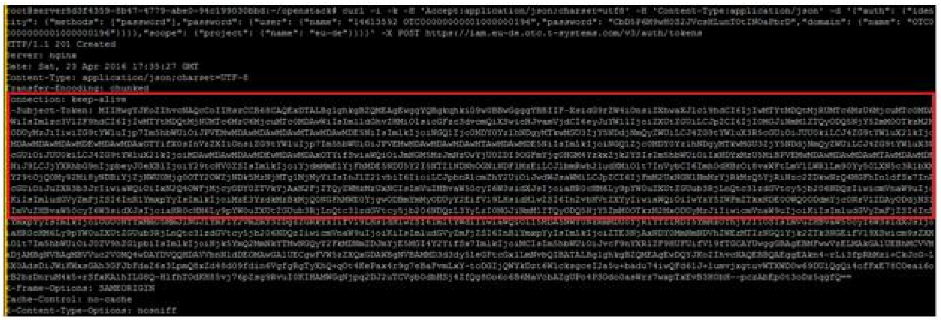
The following operations describe how to create an H2 ECS:

1. Obtain the token information.
   - URI

     POST /v3/auth/tokens
   - Example request

     ```
     curl -i -k -H 'Accept:application/json;charset=utf8' -H 'Content-Type:application/json' -d '
     {"auth": {"identity": {"methods": ["password"],"password": {"user": {"name":
     "$OS_USERNAME","password": "$OS_PASSWORD","domain":
     {"name":"$OS_USER_DOMAIN_NAME""}}}},"scope": {"project": {"name": "eu-de"}}}}' -X POST
     https://iam.ap-southeast-1.myhuaweicloud.com/v3/auth/tokens
     ```
   - Example response

Figure 2-1 Obtaining the token



```
root@server8d3f4359-8b47-4779-abe0-94c199030bbd:~/openstack# curl -i -k -H 'Accept:application/json;charset=utf8' -H 'Content-Type:application/json' -d '{"auth": {"identity": {"methods": ["password"],"password": {"user": {"name": "14613592 OTC00000000001000000194","password": "[illegible]","domain": {"name": "OTC00000000001000000194"}}}},"scope": {"project": {"name": "eu-de"}}}}' -X POST https://iam.eu-de.otc.t-systems.com/v3/auth/tokens
HTTP/1.1 201 Created
Server: nginx
Date: Sat, 23 Apr 2016 17:35:27 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Subject-Token: MIIHwgYJKoZIhvcNAQcCoIIHszCCB68CAQExDTALBglghkgBZQMEAgEwggYQBgkqhkiG9w0BBwGgggYBBIIF-[illegible]
[illegible]
X-Frame-Options: SAMEORIGIN
Cache-Control: no-cache
X-Content-Type-Options: nosniff
```

2. Create a VPC.
   - URI

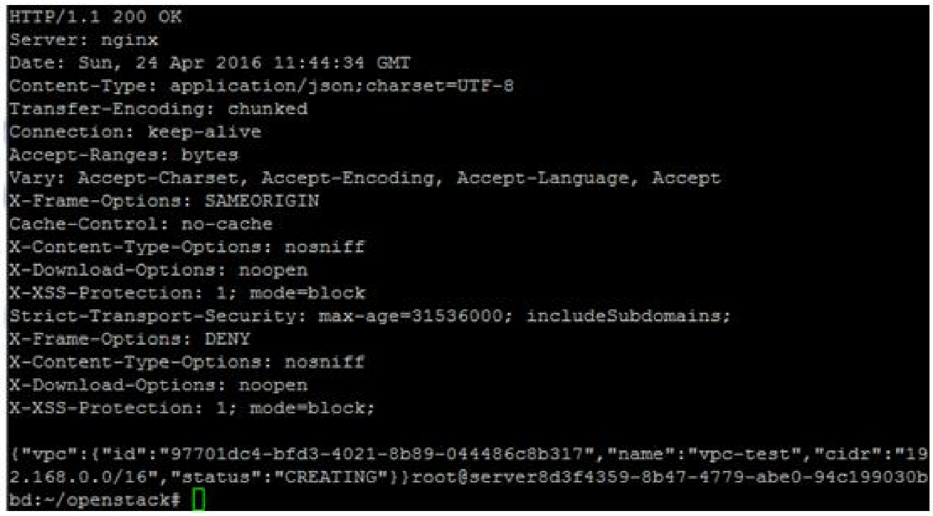
     POST /v1/{$tenant_id}/ vpcs
   - Example request

```
curl -i -k -H 'Accept:application/json;charset=utf8' -H 'Content-Type:application/json' -H "X-Auth-Token:$TOKEN " -d '
{
      "vpc": {
               "name": "vpc-test",
               "cidr": "192.168.0.0/16"
      }
}' -X POST https://iam.ap-southeast-1.myhuaweicloud.com:443/v1/{$tenant_id}/vpcs
```

   - Example response

     VPC-id: 97701dc4-bfd3-4021-8b89-044486c8b317

Figure 2-2 Creating a VPC



```
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 24 Apr 2016 11:44:34 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Accept-Ranges: bytes
Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
X-Frame-Options: SAMEORIGIN
Cache-Control: no-cache
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000; includeSubdomains;
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-XSS-Protection: 1; mode=block;

{"vpc":{"id":"97701dc4-bfd3-4021-8b89-044486c8b317","name":"vpc-test","cidr":"192.168.0.0/16","status":"CREATING"}}root@server8d3f4359-8b47-4779-abe0-94c199030bbd:~/openstack# 
```

3. This interface is used to create a subnet.
   - URI

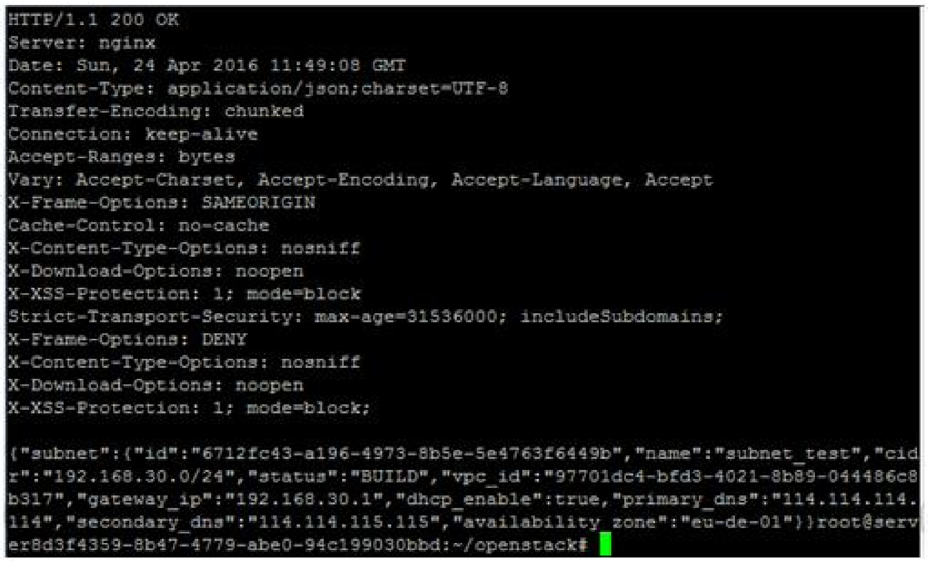
     POST /v1/{$tenant_id}/subnets
   - Example request

```
curl-i-k-H'Accept: application/json;charset=utf8'-H'Content-Type: application/json'-H"X-Auth-Token:$TOKEN "-d'{
    "subnet": {
        "name": "subnet_test",
        "cidr": "192.168.30.0/24",
        "gateway_ip": "192.168.30.1",
        "dhcp_enable": "true",
        "primary_dns": "114.114.114.114",
        "secondary_dns": "114.114.115.115",
        "availability_zone": "eu-de-01",
        "vpc_id": "97701dc4-bfd3-4021-8b89-044486c8b317"
    }
}'-XPOSThttps: //iam.ap-southeast-1.myhuaweicloud.com: 443/v1/{
    $tenant_id
}/subnets
```

– Example response

  Subnet-id: 6712fc43-a196-4973-8b5e-5e4763f6449b

Figure 2-3 Creating a subnet



```
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 24 Apr 2016 11:49:08 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Accept-Ranges: bytes
Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
X-Frame-Options: SAMEORIGIN
Cache-Control: no-cache
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000; includeSubdomains;
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-XSS-Protection: 1; mode=block;

{"subnet":{"id":"6712fc43-a196-4973-8b5e-5e4763f6449b","name":"subnet_test","cid
r":"192.168.30.0/24","status":"BUILD","vpc_id":"97701dc4-bfd3-4021-8b89-044486c8
b317","gateway_ip":"192.168.30.1","dhcp_enable":true,"primary_dns":"114.114.114.
114","secondary_dns":"114.114.115.115","availability_zone":"eu-de-01"}}root@serv
er8d3f4359-8b47-4779-abe0-94c199030bbd:~/openstack#
```

4. Create an EIP.
   – URI

     POST /v1/{$tenant_id}/publicips

   – Example request

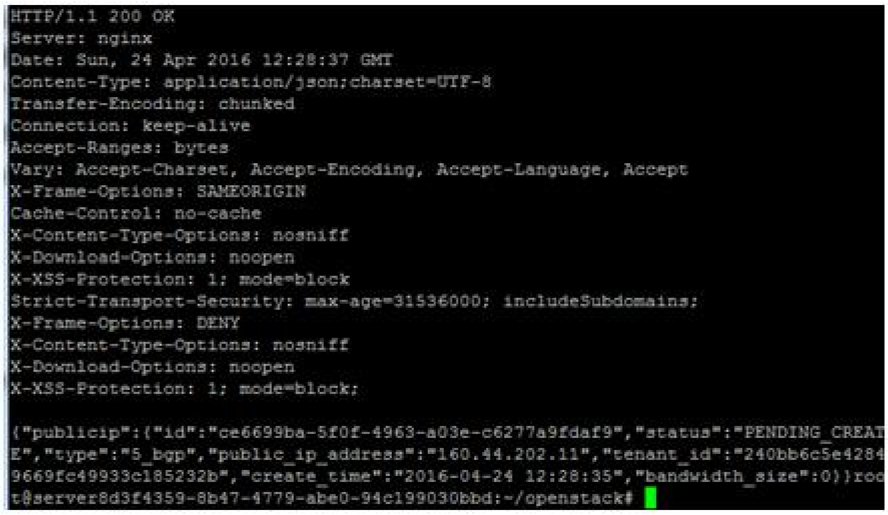
```
curl -i -k -H 'Accept:application/json;charset=utf8' -H 'Content-Type:application/json' -H 'X-Auth-Token:$TOKEN ' -d '{"publicip":{"type":"5_bgp"},"bandwidth":
{"name":"apiTest","size":111,"share_type":"PER","charge_mode":"traffic"}}' -X POST https://
iam.ap-southeast-1.myhuaweicloud.com:443/v1/{$tenant_id}/publicips
```

   – Example response

     EIP:160.44.202.11

     EIP ID: ce6699ba-5f0f-4963-a03e-c6277a9fdaf9

Figure 2-4 Creating an EIP



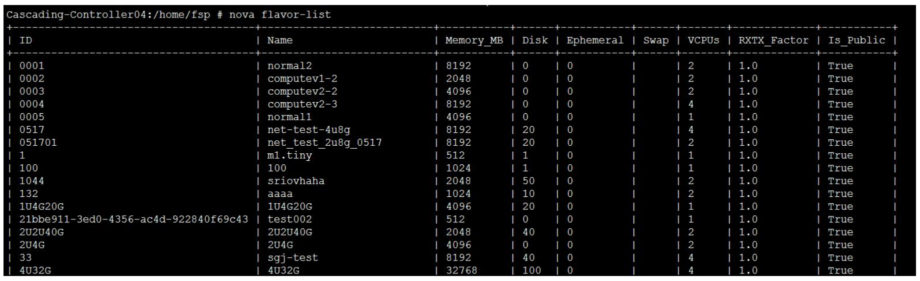
```
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 24 Apr 2016 12:28:37 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Accept-Ranges: bytes
Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
X-Frame-Options: SAMEORIGIN
Cache-Control: no-cache
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000; includeSubdomains;
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-XSS-Protection: 1; mode=block;

{"publicip":{"id":"ce6699ba-5f0f-4963-a03e-c6277a9fdaf9","status":"PENDING_CREAT
E","type":"5_bgp","public_ip_address":"160.44.202.11","tenant_id":"240bb6c5e4284
9669fc49933c185232b","create_time":"2016-04-24 12:28:35","bandwidth_size":0}}roo
t@server8d3f4359-8b47-4779-abe0-94c199030bbd:~/openstack#
```

5. Query the flavor list.
   - Using the client

     Run the following command to query the flavor list:

     nova flavor-list

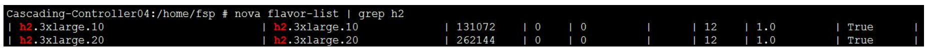
     Figure 2-5 Querying the flavor list

```
Cascading-Controller04:/home/fsp # nova flavor-list
+--------------------------------------+--------------------------+-----------+------+-----------+------+-------+-------------+-----------+
| ID                                   | Name                     | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+--------------------------------------+--------------------------+-----------+------+-----------+------+-------+-------------+-----------+
| 0001                                 | normal2                  | 8192      | 0    | 0         |      | 2     | 1.0         | True      |
| 0002                                 | computev1-2              | 2048      | 0    | 0         |      | 2     | 1.0         | True      |
| 0003                                 | computev2-2              | 4096      | 0    | 0         |      | 2     | 1.0         | True      |
| 0004                                 | computev2-3              | 8192      | 0    | 0         |      | 4     | 1.0         | True      |
| 0005                                 | normal1                  | 4096      | 0    | 0         |      | 1     | 1.0         | True      |
| 0517                                 | net-test-4u8g            | 8192      | 20   | 0         |      | 4     | 1.0         | True      |
| 051701                               | net_test_2u8g_0517       | 8192      | 20   | 0         |      | 2     | 1.0         | True      |
| 1                                    | m1.tiny                  | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 100                                  | 100                      | 1024      | 1    | 0         |      | 1     | 1.0         | True      |
| 1044                                 | sriovhaha                | 2048      | 50   | 0         |      | 2     | 1.0         | True      |
| 132                                  | aaaa                     | 1024      | 10   | 0         |      | 2     | 1.0         | True      |
| 1U4G20G                              | 1U4G20G                  | 4096      | 20   | 0         |      | 1     | 1.0         | True      |
| 21bbe911-3ed0-4356-ac4d-922840f69c43 | test002                  | 512       | 0    | 0         |      | 1     | 1.0         | True      |
| 2U2U40G                              | 2U2U40G                  | 2048      | 40   | 0         |      | 2     | 1.0         | True      |
| 2U4G                                 | 2U4G                     | 4096      | 0    | 0         |      | 2     | 1.0         | True      |
| 33                                   | sgj-test                 | 8192      | 40   | 0         |      | 4     | 1.0         | True      |
| 4U32G                                | 4U32G                    | 32768     | 100  | 0         |      | 4     | 1.0         | True      |
```

     nova flavor-list | grep h2

     Figure 2-6 Querying the H2 ECS flavor list

```
Cascading-Controller04:/home/fsp # nova flavor-list | grep h2
| h2.3xlarge.10                        | h2.3xlarge.10            | 131072    | 0    | 0         |      | 12    | 1.0         | True      |
| h2.3xlarge.20                        | h2.3xlarge.20            | 262144    | 0    | 0         |      | 12    | 1.0         | True      |
```

   - Using the **curl** command
     - URI

       GET /v2/{$tenant_id}/flavors/detail

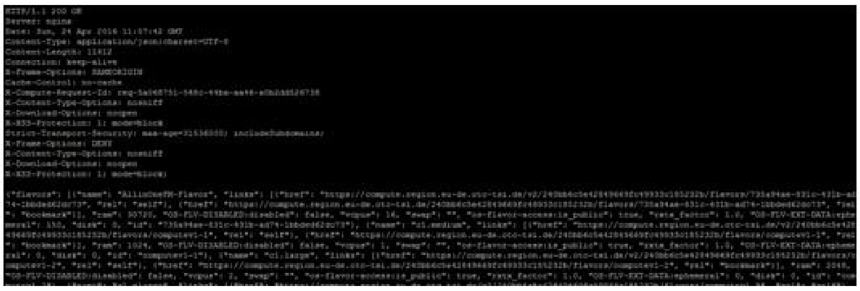
     - Example request

```
curl -g -i -X GET https://iam.ap-southeast-1.myhuaweicloud.com:443/v2/{$tenant_id}/
flavors/detail -H "User-Agent: python-novaclient" -H "Accept: application/json" -H "X-
Auth-Token: $TOKEN"
```

     - Example response

       Flavor id Example: h2.3xlarge.10

Figure 2-7 Querying the flavor list



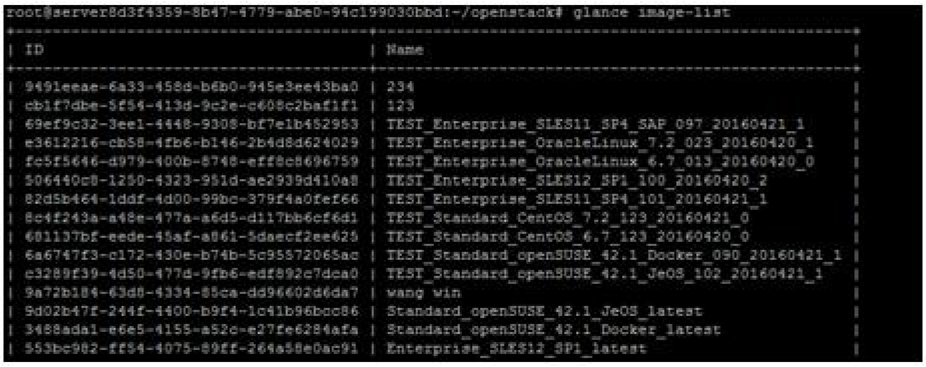
```
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 24 Apr 2016 11:57:42 GMT
Content-Type: application/json;charset=UTF-8
Content-Length: 11612
Connection: keep-alive
X-Frame-Options: SAMEORIGIN
Cache-Control: no-cache
X-Compute-Request-Id: [illegible]
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000; includeSubdomains;
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-XSS-Protection: 1; mode=block;

{"flavors": [{"name": [illegible]
```

6. Query the image list.
   - Using the client

     Run the following command to query the image list:

     glance image-list

     Figure 2-8 Querying the image list

```
root@server8d3f4359-8b47-4779-abe0-94c199030bbd:~/openstack# glance image-list
+--------------------------------------+-----------------------------------------------------+
| ID                                   | Name                                                |
+--------------------------------------+-----------------------------------------------------+
| 9491eeae-6a33-458d-b6b0-945e3ee43ba0 | 234                                                 |
| cb1f7dbe-5f54-413d-9c2e-c608c2baf1f1 | 123                                                 |
| 69ef9c32-3ee1-4448-9308-bf7e1b452953 | TEST_Enterprise_SLES11_SP4_SAP_097_20160421_1       |
| e3612216-cb58-4fb6-b146-2b4d8d624029 | TEST_Enterprise_OracleLinux_7.2_023_20160420_1      |
| fc5f5646-d979-400b-8748-eff8c8696759 | TEST_Enterprise_OracleLinux_6.7_013_20160420_0      |
| 506440c8-1250-4323-951d-ae2939d410a8 | TEST_Enterprise_SLES12_SP1_100_20160420_2           |
| 82d5b464-1ddf-4d00-99bc-379f4a0fef66 | TEST_Enterprise_SLES11_SP4_101_20160421_1           |
| 8c4f243a-a48e-477a-a6d5-d117bb6cf6d1 | TEST_Standard_CentOS_7.2_123_20160421_0             |
| 681137bf-eede-45af-a861-5daecf2ee625 | TEST_Standard_CentOS_6.7_123_20160420_0             |
| 6a6747f3-c172-430e-b74b-5c95572065ac | TEST_Standard_openSUSE_42.1_Docker_090_20160421_1 |
| c3289f39-4d50-477d-9fb6-edf892c7dca0 | TEST_Standard_openSUSE_42.1_JeOS_102_20160421_1     |
| 9a72b184-63d8-4334-85ca-dd96602d6da7 | wang win                                            |
| 9d02b47f-244f-4400-b9f4-1c41b96bcc86 | Standard_openSUSE_42.1_JeOS_latest                  |
| 3488ada1-e6e5-4155-a52c-e27fe6284afa | Standard_openSUSE_42.1_Docker_latest                |
| 553bc982-ff54-4075-89ff-264a58e0ac91 | Enterprise_SLES12_SP1_latest                        |
```

   - Using the **curl** command
     - URI

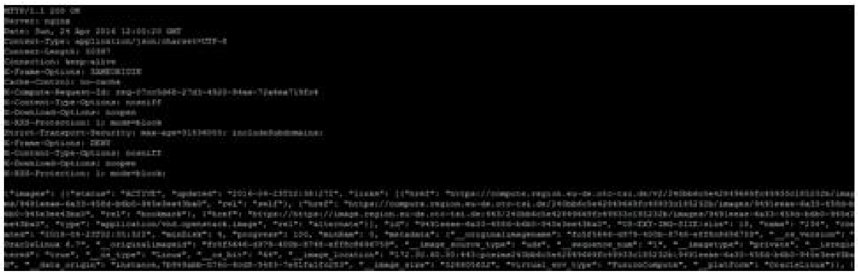
       GET /v2/{$tenant_id}/images/detail
     - Example request

```
curl -g -i -X GET https://iam.ap-southeast-1.myhuaweicloud.com:443/v2/{$tenant_id}/
images/detail -H "User-Agent: python-novaclient" -H "Accept: application/json" -H "X-
Auth-Token:$TOKEN"
```

     - Example response

       Image id Example: 7474de73-9618-4c6a-afaa-df60df57c9b9

       Figure 2-9 Querying the image list

```
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 24 Apr 2016 12:00:20 GMT
Content-Type: application/json;charset=UTF-8
Content-Length: [illegible]
Connection: keep-alive
X-Frame-Options: SAMEORIGIN
Cache-Control: no-cache
X-Compute-Request-Id: [illegible]
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000; includeSubdomains;
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-XSS-Protection: 1; mode=block;

{"images": [{"status": "ACTIVE", "updated": [illegible]
```

7. Creating an ECS.

- URI

  POST /v2/{project_id}/servers

- Example request

  ```
  curl -i -k -H 'Accept:application/json;charset=utf8' -H 'Content-Type:application/json' -H 'X-Auth-Token:$TOKEN' -d '{"server": {"availability_zone": "eu-de-01","adminPass": "Test@123","name": "h2_vm","flavorRef": "h2.3xlarge.10","networks": [{"uuid":"6712fc43-a196-4973-8b5e-5e4763f6449b"}],"imageRef":"7474de73-9618-4c6a-afaa-df60df57c9b9"}}' -X POST https://46.29.103.37:443/v2/240bb6c5e42849669fc49933c185232b/servers
  ```

- Example response

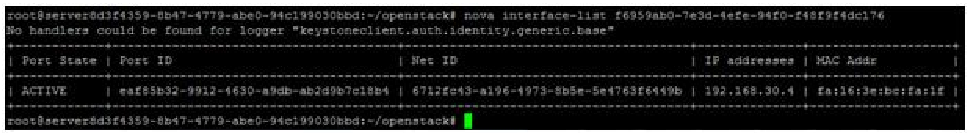
  ```
  {
      "server": {
          "security_groups": [
              {
                  "name": "default"
              }
          ],
          "OS-DCF:diskConfig": " MANUAL",
          "id": "877a2cda-ba63-4e1e-b95f-e67e48b6129a",
          "links": [
              {
                  "href": "https://46.29.103.37:443/v2/240bb6c5e42849669fc49933c185232b/servers/
  877a2cda-ba63-4e1e-b95f-e67e48b6129a",
                  "rel": "self"
              },
              {
                  "href": "http://46.29.103.37:443/240bb6c5e42849669fc49933c185232b/servers/
  877a2cda-ba63-4e1e-b95f-e67e48b6129a",
                  "rel": "bookmark"
              }
          ],
          "adminPass": "******"
      }
  }
  ```

8. Run the following command to query the NIC ID of the ECS:

   nova interface-list {$VMID}

   Information similar to the following is displayed.

   Figure 2-10 Querying the NIC ID



```
root@server8d3f4359-8b47-4779-abe0-94c199030bbd:~/openstack# nova interface-list f6959ab0-7e3d-4efe-94f0-f48f9f4dc176
No handlers could be found for logger "keystoneclient.auth.identity.generic.base"
+------------+--------------------------------------+--------------------------------------+---------------+-------------------+
| Port State | Port ID                              | Net ID                               | IP addresses  | MAC Addr          |
+------------+--------------------------------------+--------------------------------------+---------------+-------------------+
| ACTIVE     | eaf85b32-9912-4630-a9db-ab2d9b7c18b4 | 6712fc43-a196-4973-8b5e-5e4763f6449b | 192.168.30.4  | fa:16:3e:bc:fa:1f |
+------------+--------------------------------------+--------------------------------------+---------------+-------------------+
root@server8d3f4359-8b47-4779-abe0-94c199030bbd:~/openstack#
```

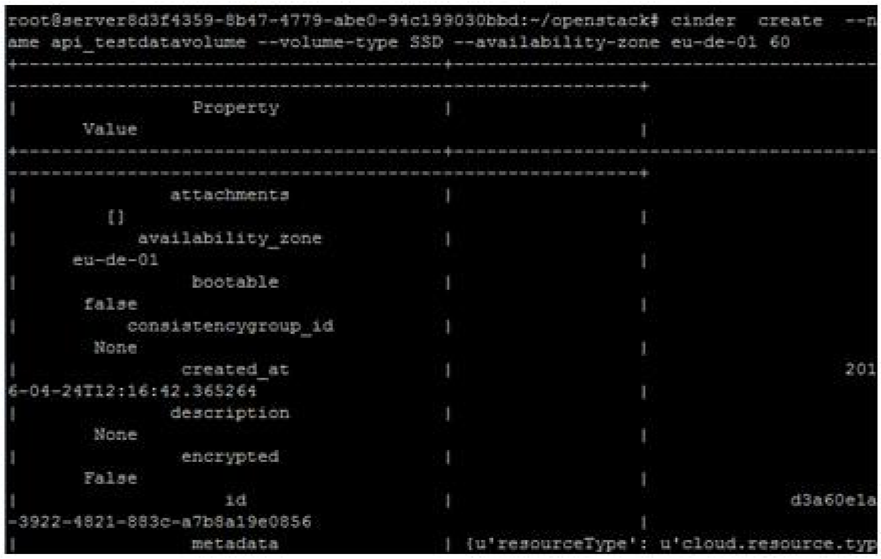
   The NIC ID is **Vmid= eaf85b32-9912-4630-a9db-ab2d9b7c18b4**.

9. Run the following command to create a data disk:

   cinder create --name datavolume --volume-type SATA --availability-zone eu-de-01 60

   Information similar to the following is displayed.

Figure 2-11 Creating a data disk



```
root@server8d3f4359-8b47-4779-abe0-94c199030bbd:~/openstack# cinder  create  --n
ame api_testdatavolume --volume-type SSD --availability-zone eu-de-01 60
+--------------------------------------+-----------------------------------------
-------------------------------------------------------+
|               Property               |
      Value                                            |
+--------------------------------------+-----------------------------------------
-------------------------------------------------------+
|             attachments              |
         []                                            |
|          availability_zone           |
     eu-de-01                                          |
|               bootable               |
       false                                           |
|         consistencygroup_id          |
        None                                           |
|              created_at              |                                      201
6-04-24T12:16:42.365264                                |
|             description              |
        None                                           |
|              encrypted               |
       False                                           |
|                  id                  |                                 d3a60e1a
-3922-4821-883c-a7b8a19e0856                           |
|               metadata               | {u'resourceType': u'cloud.resource.typ
```

The data disk ID is **Datadiskid= d3a60e1a-3922-4821-883c-a7b8a19e0856**.

10. Run the following command to check the data disk status:

    cinder show {volumeId}

    If the data disk status is available, you can attach it to the ECS.

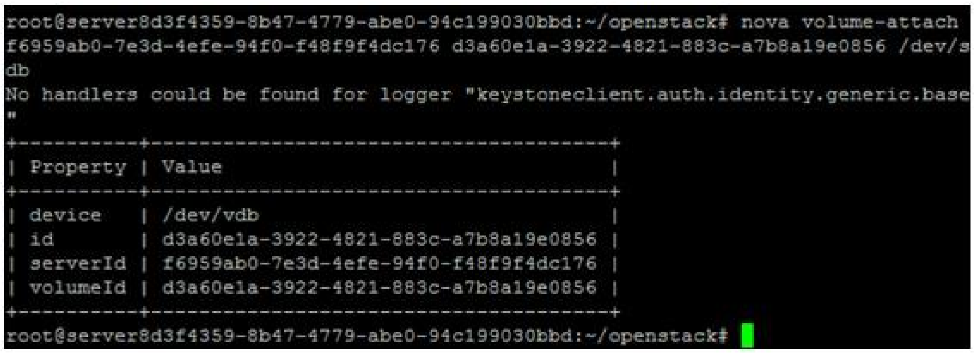
11. Run the following command to attach the data disk to the ECS:

    nova volume-attach {serverId} {volumeId} device_name

    An example command is as follows:

    nova volume-attach f6959ab0-7e3d-4efe-94f0-f48f9f4dc176 d3a60e1a-3922-4821-883c-a7b8a19e0856 /dev/sdb

Figure 2-12 Attaching a data disk



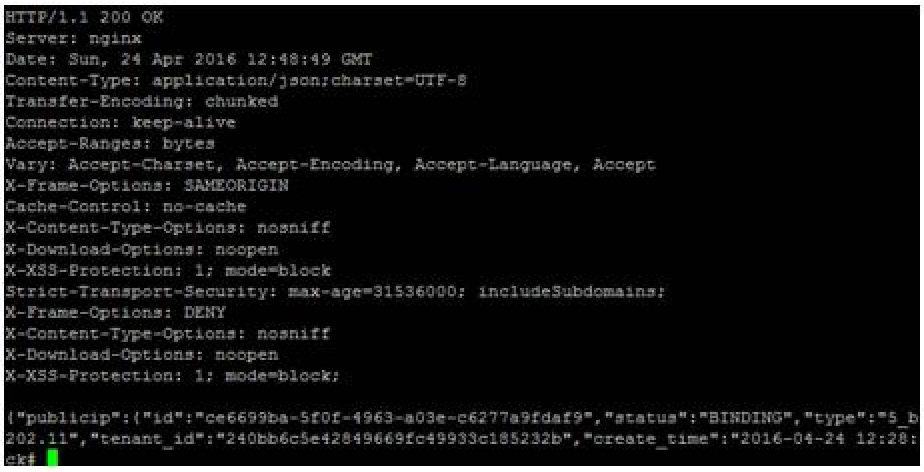
```
root@server8d3f4359-8b47-4779-abe0-94c199030bbd:~/openstack# nova volume-attach
f6959ab0-7e3d-4efe-94f0-f48f9f4dc176 d3a60e1a-3922-4821-883c-a7b8a19e0856 /dev/s
db
No handlers could be found for logger "keystoneclient.auth.identity.generic.base
"
+----------+--------------------------------------+
| Property | Value                                |
+----------+--------------------------------------+
| device   | /dev/vdb                             |
| id       | d3a60e1a-3922-4821-883c-a7b8a19e0856 |
| serverId | f6959ab0-7e3d-4efe-94f0-f48f9f4dc176 |
| volumeId | d3a60e1a-3922-4821-883c-a7b8a19e0856 |
+----------+--------------------------------------+
root@server8d3f4359-8b47-4779-abe0-94c199030bbd:~/openstack# 
```

12. Bind an EIP.
    - URI

      PUT /v1/{$tenant_id}/publicips/{EIPid}
    - Example request

      ```
      curl -i -k -H 'Accept:application/json;charset=utf8' -H 'Content-Type:application/json' -H 'X-Auth-Token:$TOKEN' -d '{"publicip":{"port_id":"eaf85b32-9912-4630-a9db-ab2d9b7c18b4"}}' -X PUT https://46.29.103.37:443/v1/{$tenant_id}/publicips/ce6699ba-5f0f-4963-a03e-c6277a9fdaf9
      ```
    - Example response

Figure 2-13 Binding an EIP



```
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 24 Apr 2016 12:48:49 GMT
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Accept-Ranges: bytes
Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
X-Frame-Options: SAMEORIGIN
Cache-Control: no-cache
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000; includeSubdomains;
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-Download-Options: noopen
X-XSS-Protection: 1; mode=block;

{"publicip":{"id":"ce6699ba-5f0f-4963-a03e-c6277a9fdaf9","status":"BINDING","type":"5_b
202.11","tenant_id":"240bb6c5e42849669fc49933c185232b","create_time":"2016-04-24 12:28:
ck#
```

2.2 Configuring Password-free Login to an ECS

Scenarios

This section describes how to configure an ECS to enable password-free login to it.

Background Information

$: indicates performing an operation as a common user.

#: indicates performing an operation as an administrator.

Run the **sudo su** command to switch from a common user to an administrator.

Prerequisites

An ECS has been created successfully and an EIP has been bound to it.

Procedure

1. Use PuTTY and a key pair to log in to an ECS in the cluster.
2. Run the following command to disable user logout upon system timeout:

   # TMOUT=0
3. Copy the ECS private key file (for example ***.pem**) to the **.ssh** directory and name it as **id_rsa**.

   $ cd ~/.ssh

   $ mv *.pem id_rsa
4. Run the following command to configure permissions of the key file:

   $ sudo chmod 600 id_rsa
5. Run the following command to query the host name:

   # hostname
6. Run the following command to add the private network IP address and name of the host:

# vi /etc/hosts

An example command is as follows:

192.168.0.1 ecs-ff-0001

7. Run the following commands to log in to the node using SSH and check whether you can log in to the ECS without a password:

   For example, if the hostname is **hostname1**, run the following commands:

   $ ssh localhost

   $ ssh *hostname1*

2.3 Installing and Using MPIs

2.3.1 MPIs Supported in the ECS Scenario

The following MPIs are supported in the ECS scenario:

- Built-in OpenMPI of the IB driver
- Community OpenMPI
- Spectrum MPI
- Intel MPI
- Platform MPI

Install and use an MPI as needed.

2.3.2 Built-in OpenMPI of the IB Driver

Scenarios

This section describes how to install and use the built-in OpenMPI of the IB driver (for example, 3.0.0rc6).

Prerequisites

You have configured the ECS password-free login.

Procedure

Step 1 Check whether the IB driver has been installed.

1. Use PuTTY and a key pair to log in to the ECS.
2. Run the following command to switch to user **root**:

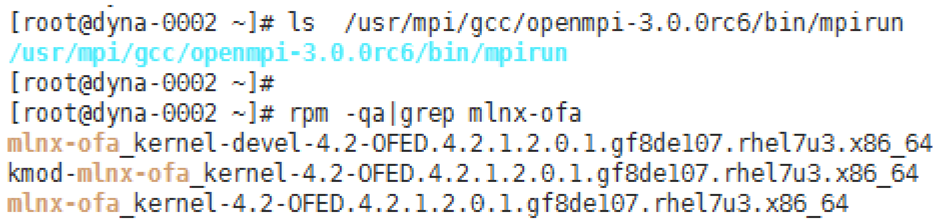
   $ sudo su
3. Run the following command to disable user logout upon system timeout:

   # TMOUT=0
4. Run the following commands to check whether the IB driver has been installed:

   # rpm -qa | grep mlnx-ofa

   # ls /usr/mpi/gcc/openmpi-3.0.0rc6/bin/mpirun

Figure 2-14 Command output indicating that the IB driver has been installed



```
[root@dyna-0002 ~]# ls  /usr/mpi/gcc/openmpi-3.0.0rc6/bin/mpirun
/usr/mpi/gcc/openmpi-3.0.0rc6/bin/mpirun
[root@dyna-0002 ~]#
[root@dyna-0002 ~]# rpm -qa|grep mlnx-ofa
mlnx-ofa_kernel-devel-4.2-OFED.4.2.1.2.0.1.gf8de107.rhel7u3.x86_64
kmod-mlnx-ofa_kernel-4.2-OFED.4.2.1.2.0.1.gf8de107.rhel7u3.x86_64
mlnx-ofa_kernel-4.2-OFED.4.2.1.2.0.1.gf8de107.rhel7u3.x86_64
```

- If the preceding two commands contain the returned value shown in **Figure 2-14**, the IB driver has been installed. Go to **Step 3**.
- If the returned value is different from that shown in **Figure 2-14**, the IB driver is not installed. Go to **Step 2**.

Step 2 Download and install the IB driver.

Download the required version of InfiniBand NIC driver from the Mellanox official website **https://network.nvidia.com/products/infiniband-drivers/linux/mlnx_ofed/** and install the driver by following the instructions provided.

For example, for an ECS running CentOS 7.3, download the **MLNX_OFED_LINUX-4.2-1.2.0.0-rhel7.3-x86_64.tgz** installation package and run the following commands to install the IB driver:

# yum install tk tcl

# tar -xvf MLNX_OFED_LINUX-4.2-1.2.0.0-rhel7.3-x86_64.tgz

# cd MLNX_OFED_LINUX-4.2-1.2.0.0-rhel7.3-x86_64/

# ./mlnxofedinstall

Step 3 Configure environment variables.

1. Run the following commands to configure the **~/.bashrc** file using the vim editor and add the following content:

   export PATH=$PATH:/usr/mpi/gcc/openmpi-3.0.0rc6/bin

   export LD_LIBRARY_PATH=/usr/mpi/gcc/openmpi-3.0.0rc6/lib64

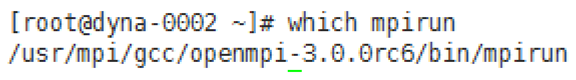
2. Run the following command to import MPI environment variables:

   # source ~/.bashrc

3. Run the following command to check whether the MPI environment variables are correct:

   # which mpirun

   Figure 2-15 Checking MPI environment variables

   ```
   [root@dyna-0002 ~]# which mpirun
   /usr/mpi/gcc/openmpi-3.0.0rc6/bin/mpirun
   ```

   If information shown in **Figure 2-15** is displayed, the environment configuration is correct.

Step 4 Run the following command to run Intel MPI benchmark on an ECS:

# mpirun --allow-run-as-root -np 2 /usr/mpi/gcc/openmpi-3.0.0rc6/tests/imb/IMB-MPI1 PingPong

Information similar to the following is displayed:

```
#------------------------------------------------------------
#    Intel (R) MPI Benchmarks 4.1, MPI-1 part
#------------------------------------------------------------
# Date                  : Mon Jul 16 10:11:14 2018
# Machine               : x86_64
# System                : Linux
# Release               : 3.10.0-514.10.2.el7.x86_64
# Version               : #1 SMP Fri Mar 3 00:04:05 UTC 2017
# MPI Version           : 3.1
# MPI Thread Environment:

# New default behavior from Version 3.2 on:

# the number of iterations per message size is cut down
# dynamically when a certain run time (per message size sample)
# is expected to be exceeded. Time limit is defined by variable
# "SECS_PER_SAMPLE" (=> IMB_settings.h)
# or through the flag => -time


# Calling sequence was:

# /usr/mpi/gcc/openmpi-3.0.0rc6/tests/imb/IMB-MPI1 PingPong

# Minimum message length in bytes:   0
# Maximum message length in bytes:   4194304
#
# MPI_Datatype                   :   MPI_BYTE
# MPI_Datatype for reductions    :   MPI_FLOAT
# MPI_Op                         :   MPI_SUM
#
#

# List of Benchmarks to run:

# PingPong

#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
#bytes #repetitions      t[usec]   Mbytes/sec
0         1000         0.24         0.00
1         1000         0.25         3.89
2         1000         0.23         8.17
4         1000         0.23        16.25
8         1000         0.23        32.48
16         1000         0.23        65.98
32         1000         0.26       115.35
64         1000         0.26       232.92
128         1000         0.38       320.59
256         1000         0.44       554.35
512         1000         0.54       902.98
1024         1000         0.64      1537.63
2048         1000         0.85      2298.79
4096         1000         1.28      3057.93
8192         1000         2.28      3426.14
16384         1000         1.41     11052.14
32768         1000         2.05     15218.39
65536          640         3.31     18882.34
131072          320         6.57     19036.27
262144          160        15.12     16535.96
524288           80        32.90     15195.74
1048576           40        64.62     15476.02
```

```
2097152         20      122.83    16282.06
4194304         10      242.95    16463.95

# All processes entering MPI_Finalize
```

----End

2.3.3 Community Open MPI

Scenarios

This section describes how to install and use community Open MPI (for example, version 3.1.1).

Prerequisites

You have configured the ECS password-free login.

Procedure

Step 1 Install the HPC-X toolkit.

1. Download the desired HPC-X toolkit and Open MPI.

   To use the community Open MPI, you must use the Mellanox HPC-X toolkit. Download the desired version of the HPC-X toolkit based on the ECS OS and IB driver versions. An example of the HPC-X toolkit version is **hpcx-v2.0.0-gcc-MLNX_OFED_LINUX-4.2-1.2.0.0-redhat7.3-x86_64.tbz**.

2. Run the following command to decompress the HPC-X toolkit:

   # tar -xvf hpcx-v2.0.0-gcc-MLNX_OFED_LINUX-4.2-1.2.0.0-redhat7.3-x86_64.tbz

3. (Optional) Run the following command to change the directory of the HPC-X toolkit:

   # mv hpcx-v2.0.0-gcc-MLNX_OFED_LINUX-4.2-1.2.0.0-redhat7.3-x86_64.tbz /opt/hpcx-v2.0.0

Step 2 Install Open MPI.

1. Copy the Open MPI package (for example, **openmpi-3.1.1.tar.gz**) to the ECS and run the following commands to decompress it:

   # tar -xzvf openmpi-3.1.1.tar.gz

   # cd openmpi-3.1.1

2. Run the following command to install the required library file:

   # yum install binutils-devel.x86_64 libibverbs-devel

3. Run the following commands to compile and install Open MPI:

   # ./autogen.pl

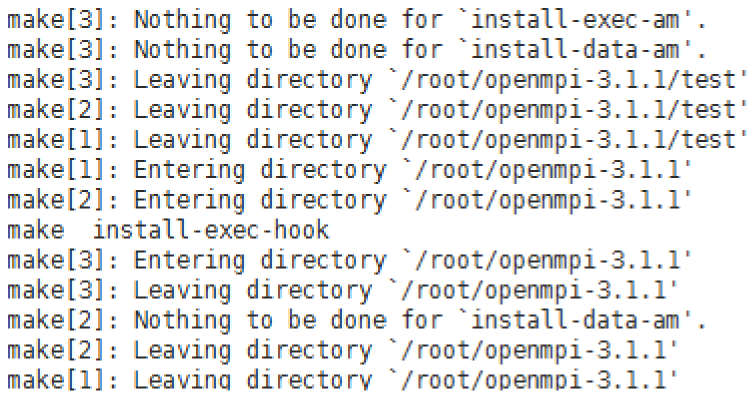
   # mkdir build

   # cd build

   # ../configure --prefix=/opt/openmpi-311 --with-mxm=/opt/hpcx-v2.0.0/mxm

   # make all install

Figure 2-16 Installing Open MPI



```
make[3]: Nothing to be done for `install-exec-am'.
make[3]: Nothing to be done for `install-data-am'.
make[3]: Leaving directory `/root/openmpi-3.1.1/test'
make[2]: Leaving directory `/root/openmpi-3.1.1/test'
make[1]: Leaving directory `/root/openmpi-3.1.1/test'
make[1]: Entering directory `/root/openmpi-3.1.1'
make[2]: Entering directory `/root/openmpi-3.1.1'
make  install-exec-hook
make[3]: Entering directory `/root/openmpi-3.1.1'
make[3]: Leaving directory `/root/openmpi-3.1.1'
make[2]: Nothing to be done for `install-data-am'.
make[2]: Leaving directory `/root/openmpi-3.1.1'
make[1]: Leaving directory `/root/openmpi-3.1.1'
```

If information shown in **Figure 2-16** is displayed and no error is displayed, the installation is successful.

Step 3 Configure MPI environment variables.

1. Add the following environment variables to **~/.bashrc**:

   export PATH=$PATH:/opt/openmpi-311/bin

   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/openmpi-311/lib

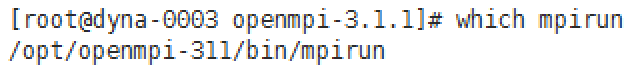
2. Run the following command to import MPI environment variables:

   # source ~/.bashrc

3. Run the following command to check whether the MPI environment variables are correct:

   # which mpirun

Figure 2-17 Checking community Open MPI environment variables



```
[root@dyna-0003 openmpi-3.1.1]# which mpirun
/opt/openmpi-311/bin/mpirun
```

If information shown in **Figure 2-17** is displayed, the environment configuration is correct.

Step 4 Run the following command to run Intel MPI benchmark on an ECS:

$ mpirun --allow-run-as-root -np 2 /usr/mpi/gcc/openmpi-3.0.0rc6/tests/imb/IMB-MPI1 PingPong

Information similar to the following is displayed:

```
#------------------------------------------------------------
#    Intel (R) MPI Benchmarks 4.1, MPI-1 part
#------------------------------------------------------------
# Date                  : Mon Jul 16 09:38:20 2018
# Machine               : x86_64
# System                : Linux
# Release               : 3.10.0-514.10.2.el7.x86_64
# Version               : #1 SMP Fri Mar 3 00:04:05 UTC 2017
# MPI Version           : 3.1
# MPI Thread Environment:

# New default behavior from Version 3.2 on:

# the number of iterations per message size is cut down
# dynamically when a certain run time (per message size sample)
# is expected to be exceeded. Time limit is defined by variable
# "SECS_PER_SAMPLE" (=> IMB_settings.h)
# or through the flag => -time

# Calling sequence was:

# /usr/mpi/gcc/openmpi-3.0.0rc6/tests/imb/IMB-MPI1 PingPong

# Minimum message length in bytes:   0
# Maximum message length in bytes:   4194304
#
# MPI_Datatype                   :   MPI_BYTE
# MPI_Datatype for reductions    :   MPI_FLOAT
# MPI_Op                         :   MPI_SUM
#
#

# List of Benchmarks to run:

# PingPong

#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
#bytes #repetitions      t[usec]   Mbytes/sec
0         1000         0.23         0.00
1         1000         0.23         4.06
2         1000         0.24         8.04
4         1000         0.24        16.19
8         1000         0.24        32.29
16         1000         0.24        64.06
32         1000         0.27       114.46
64         1000         0.27       229.02
128         1000         0.37       333.48
256         1000         0.46       535.83
512         1000         0.52       944.51
1024         1000         0.63      1556.77
2048         1000         0.83      2349.92
4096         1000         1.35      2896.07
8192         1000         2.29      3415.98
16384         1000         1.46     10727.65
32768         1000         2.08     15037.62
65536          640         3.53     17691.38
131072          320         6.52     19159.59
262144          160        15.62     16002.93
524288           80        31.37     15938.06
1048576           40        61.78     16185.93
2097152           20       124.04     16124.41
4194304           10       242.42     16500.33

# All processes entering MPI_Finalize
```

----End

2.3.4 Spectrum MPI

Scenarios

This section describes how to install and use IBM Spectrum MPI (for example, IBM Spectrum MPI v10.1).

IBM Spectrum MPI v10.1 supports the following OSs:

- **IBM Spectrum MPI 10.1.0.1 Eval for x86_64 Linux**

- Red Hat Enterprise Linux version 6.6 and later
- Red Hat Enterprise Linux version 7.1 and later
- SUSE Linux Enterprise Server version 11 SP4
- SUSE Linux Enterprise Server version 12 and later

- **IBM Spectrum MPI 10.1.0.2 Eval for Power 8 Linux**
  - Red Hat Enterprise Linux version 7.3 and later

Prerequisites

You have configured the ECS password-free login.

Procedure

Step 1 Obtain the software package.

1. Download the IBM Spectrum MPI software package from the following website:

   Download path: **https://www-01.ibm.com/marketing/iwm/iwm/web/preLogin.do?source=swerpsysz-lsf-3**

   The software package contains the license and software packages. The following are examples:

   smpi_lic_s-10.1Eval-rh7_Sep15.x86_64.rpm

   ibm_smpi-10.1.0.3eval_170901-rh7_Apr11.x86_64.rpm

2. Download the desired HPC-X toolkit.

   In the EDR SR-IOV scenario, the IBM MPI relies on the MXM library provided by the HPC-X toolkit. Download the desired version of the HPC-X toolkit based on the ECS OS and IB driver versions. An example of the HPC-X toolkit version is **hpcx-v2.0.0-gcc-MLNX_OFED_LINUX-4.2-1.2.0.0-redhat7.3-x86_64.tbz**.

   Download path: **https://developer.nvidia.com/networking/hpc-x**

Step 2 Install the HPC-X toolkit.

1. Upload the HPC-X package downloaded in **Step 1** to the ECS with an MPI.
2. Run the following command to decompress the HPC-X toolkit:

   ```
   $ tar xvf hpcx-v2.0.0-gcc-MLNX_OFED_LINUX-4.2-1.2.0.0-redhat7.3-x86_64.tbz
   ```

3. Run the following command to configure the HPC-X environment variables:

   ```
   $ cd hpcx-v2.0.0-gcc-MLNX_OFED_LINUX-4.2-1.2.0.0-redhat7.3-x86_64
   $ export HPCX_HOME=$PWD
   ```

Step 3 Install IBM Spectrum MPI.

1. Upload the MPI package downloaded in **Step 1** to the ECS with an MPI.
2. Run the following command to switch to user **root**:

   ```
   $ sudo su -
   ```

3. Run the following command to configure environment variables:
   - If you choose to automatically accept the IBM Spectrum MPI installation license agreement, run the following command:

# export IBM_SPECTRUM_MPI_LICENSE_ACCEPT=yes

– If you choose to manually accept the IBM Spectrum MPI installation license agreement, run the following command:

# export IBM_SPECTRUM_MPI_LICENSE_ACCEPT=no

4. Install the license.

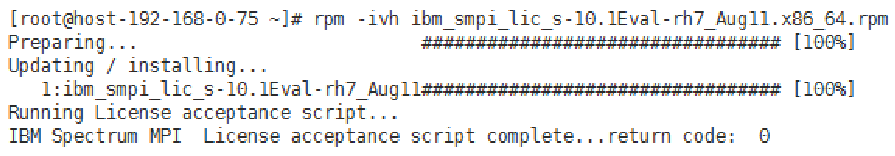
– If you choose to automatically accept the IBM Spectrum MPI installation license agreement, run the following command:

# rpm -ivh smpi_lic_s-10.1Eval-rh7_Sep15.x86_64.rpm

– If you choose to manually accept the IBM Spectrum MPI installation license agreement, run the following command:

# rpm -ivh ibm_smpi_lic_s-10.1Eval-rh7_Sep15.x86_64.rpm

Figure 2-18 Manually accepting the IBM Spectrum MPI installation license agreement



```
[root@host-192-168-0-75 ~]# rpm -ivh ibm_smpi_lic_s-10.1Eval-rh7_Aug11.x86_64.rpm
Preparing...                          ################################# [100%]
Updating / installing...
   1:ibm_smpi_lic_s-10.1Eval-rh7_Aug11################################# [100%]
Running License acceptance script...
IBM Spectrum MPI  License acceptance script complete...return code:  0
```

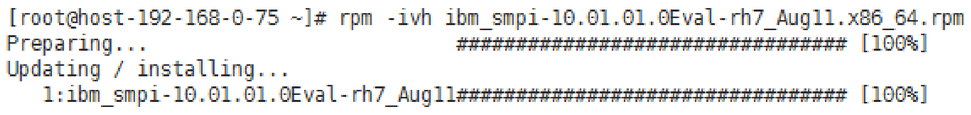
Run the following command as prompted:

# sh /opt/ibm/spectrum_mpi/lap_se/bin/accept_spectrum_mpi_license.sh

5. Run the following command to install the software:

# rpm -ivh ibm_smpi-10.1.0.3eval_170901-rh7_Apr11.x86_64.rpm

Figure 2-19 Installing the software



```
[root@host-192-168-0-75 ~]# rpm -ivh ibm_smpi-10.01.01.0Eval-rh7_Aug11.x86_64.rpm
Preparing...                          ################################# [100%]
Updating / installing...
   1:ibm_smpi-10.01.01.0Eval-rh7_Aug11################################# [100%]
```

Step 4 Configure MPI environment variables.

1. By default, Spectrum MPI is installed in the **/opt/ibm/spectrum_mpi** directory. In this case, you need to configure the following environment variables:

export MPI_ROOT=/opt/ibm/spectrum_mpi

export LD_LIBRARY_PATH=$MPI_ROOT/lib:$LD_LIBRARY_PATH

export PATH=$MPI_ROOT/bin:$PATH

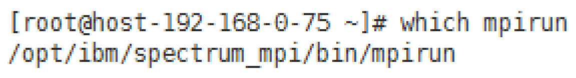
export MANPATH=$MPI_ROOT/share/man:$MANPATH

unset MPI_REMSH

2. Run the following command to check whether the environment variables have been imported:

# which mpirun

Figure 2-20 Checking MPI environment variables



```
[root@host-192-168-0-75 ~]# which mpirun
/opt/ibm/spectrum_mpi/bin/mpirun
```

Step 5 Run the following command on an ECS to run the executable file through Spectrum MPI:

1. Run the following command to edit the file:

   # cd

   # vi hello.c

   Edit the following content:

```
#include<mpi.h>
#include<stdio.h>
int main(intargc, char**argv){
//Initialize the MPI environment
MPI_Init(NULL, NULL);
//Get the number of processes
int world_size;
MPI_Comm_size(MPI_COMM_WORLD, &world_size);
//Get the rank of the process
int world_rank;
MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
//Get the name of the processor
char processor_name[MPI_MAX_PROCESSOR_NAME];
int name_len;
MPI_Get_processor_name(processor_name, &name_len);
//Print off a hello world message
printf("Hello world from processor %s, rank %d"" out of %d processors\n",      processor_name,
world_rank, world_size);
//FinalizetheMPIenvironment.
MPI_Finalize();
}
```

2. Run the following command to generate an executable file (for example in the **/home/linux** directory).

   # mpicc hello.c -o spe_hello

   NOTE

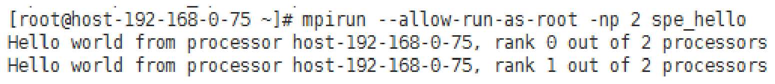
   The **hello** file varies depending on the MPI version. If you update the MPI version, run the **# mpicc hello.c -o spe_hello** command to generate a new executable file.

3. Run the following command to run Spectrum MPI on an ECS:

   # mpirun --allow-run-as-root -np 2 /root/spe_hello

   Information shown in **Figure 2-21** is displayed.

Figure 2-21 Command output



```
[root@host-192-168-0-75 ~]# mpirun --allow-run-as-root -np 2 spe_hello
Hello world from processor host-192-168-0-75, rank 0 out of 2 processors
Hello world from processor host-192-168-0-75, rank 1 out of 2 processors
```

----End

2.3.5 Intel MPI

Scenarios

This section describes how to install and use Intel MPI (for example, l_mpi_2018.0.128) on an ECS.

Prerequisites

You have configured the ECS password-free login.

Procedure

Step 1 Install Intel MPI.

1. Download Intel MPI.

   Download path: **https://software.intel.com/en-us/intel-mpi-library**

2. Run the following commands to decompress and install Intel MPI:

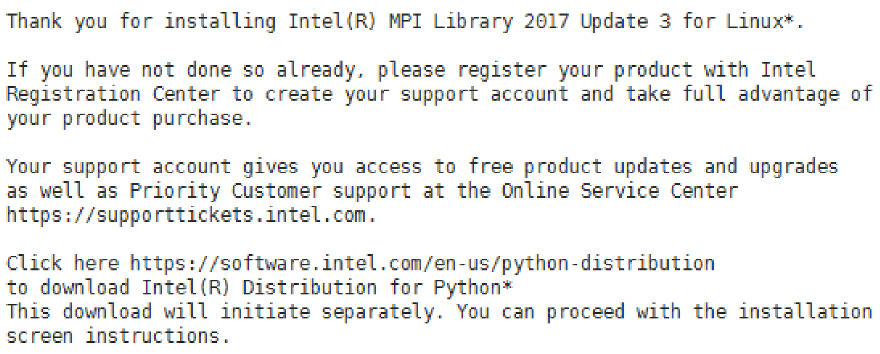
   For example, run the following commands to decompress and install **l_mpi_2018.0.128.tgz**:

   # tar -xvf l_mpi_2018.0.128.tgz

   # cd l_mpi_2018.0.128/

   # ./install.sh

   Figure 2-22 Successful Intel MPI installation



```
Thank you for installing Intel(R) MPI Library 2017 Update 3 for Linux*.

If you have not done so already, please register your product with Intel
Registration Center to create your support account and take full advantage of
your product purchase.

Your support account gives you access to free product updates and upgrades
as well as Priority Customer support at the Online Service Center
https://supporttickets.intel.com.

Click here https://software.intel.com/en-us/python-distribution
to download Intel(R) Distribution for Python*
This download will initiate separately. You can proceed with the installation
screen instructions.
```

Step 2 Configure environment variables.

1. Add the following statements in **~/.bashrc** as a common user:

   export PATH=$PATH:/opt/intel/impi/2018.0.128/bin64

   export LD_LIBRARY_PATH=/opt/intel/impi/2018.0.128/lib64

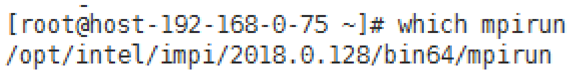
2. Run the following command to import environment variables:

   # source ~/.bashrc

Step 3 Run the following command to check whether environment variables are imported successfully:

# which mpirun

Figure 2-23 Successful importing of environment variables



```
[root@host-192-168-0-75 ~]# which mpirun
/opt/intel/impi/2018.0.128/bin64/mpirun
```

If information shown in **Figure 2-23** is displayed, the environment variables have been imported.

Step 4 Run the following commands to run Intel MPI on an ECS:

1. Run the following commands to generate an executable file:

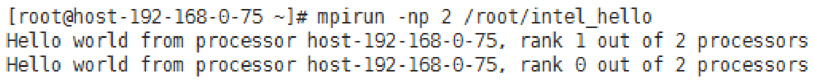
   # cd

   # mpicc hello.c -o intel_hello

2. Run the following command to run Intel MPI on an ECS:

   # mpirun -np 2 /root/intel_hello

   Figure 2-24 Running Intel MPI on an ECS

```
[root@host-192-168-0-75 ~]# mpirun -np 2 /root/intel_hello
Hello world from processor host-192-168-0-75, rank 1 out of 2 processors
Hello world from processor host-192-168-0-75, rank 0 out of 2 processors
```

----End

2.3.6 Platform MPI

Scenarios

This section describes how to install and use Platform MPI (for example, platform_mpi-09.01.04.03r-ce) on an ECS.

Prerequisites

You have configured the ECS password-free login.

Procedure

Step 1 Install Platform MPI.

1. Run the following command to install the required library file:

   # yum install glibc.i686 libgcc-4.8.5-11.el7.i686

2. Add the execution permission. For example, the installation package is stored in **/root**.

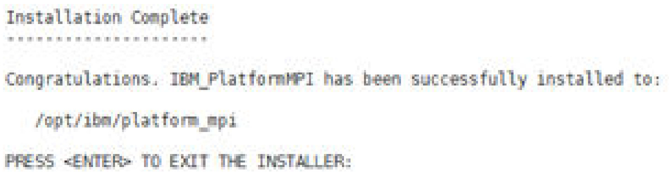
   # cd /root && chmod +x platform_mpi- 09.01.04.03r-ce.bin

3. Run the following command to install Platform MPI:

   # **./platform_mpi- 09.01.04.03r-ce.bin**

   Press **Enter** or **1** (accept the agreement) as prompted until the installation is complete.

   Figure 2-25 Successful Platform MPI installation

```
Installation Complete
---------------------

Congratulations. IBM_PlatformMPI has been successfully installed to:

   /opt/ibm/platform_mpi

PRESS <ENTER> TO EXIT THE INSTALLER:
```

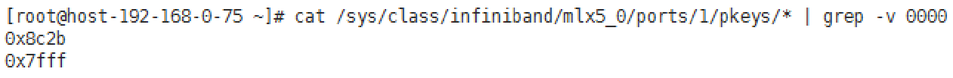
   The default installation path is **/opt/ibm/platform_mpi**.

Step 2 Configure MPI environment variables.

1. Run the following command to obtain the pkey:

   # cat /sys/class/infiniband/mlx5_0/ports/1/pkeys/* | grep -v 0000

   Figure 2-26 Obtaining the pkey

```
[root@host-192-168-0-75 ~]# cat /sys/class/infiniband/mlx5_0/ports/1/pkeys/* | grep -v 0000
0x8c2b
0x7fff
```

2. Add the following statements in **~/.bashrc** as a common user:

   export MPI_ROOT=/opt/ibm/platform_mpi

   export PATH=$MPI_ROOT/bin:$PATH

   export LD_LIBRARY_PATH=/opt/ibm/platform_mpi/lib/linux_amd64

   export MPI_IB_PKEY=*pkey obtained in* ***Step 2.1***

   $source ~/.bashrc

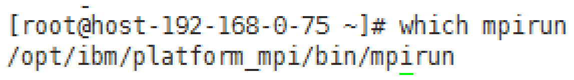
   NOTE

   If there are multiple pkeys, use a comma to separate them.

3. Run the following command to check whether the environment variables have been imported:

   # which mpirun

   Figure 2-27 Successfully importing environment variables of Platform MPI

```
[root@host-192-168-0-75 ~]# which mpirun
/opt/ibm/platform_mpi/bin/mpirun
```

Step 3 Run Platform MPI on an ECS:

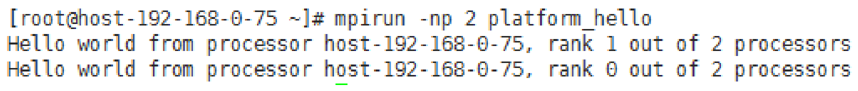
1. Run the following command to re-compile the **hello.c** file:

   # mpicc hello.c -o platform_hello

2. Run the following command to run Platform MPI on an ECS:

   # mpirun -np 2 /root/platform_hello

   Figure 2-28 Running Platform MPI on an ECS

```
[root@host-192-168-0-75 ~]# mpirun -np 2 platform_hello
Hello world from processor host-192-168-0-75, rank 1 out of 2 processors
Hello world from processor host-192-168-0-75, rank 0 out of 2 processors
```

----End

2.4 Creating a Private Image Using an ECS

Scenarios

You can use an ECS for which HPC has been configured as a template to create private images which can be used to quickly create clusters. This section describes how to convert a Linux ECS into a private image on the management console or by calling HTTPS-based APIs.

Prerequisites

- The IP address obtaining mode of the Linux ECS NIC has been set to DHCP.
- The udev rules on the Linux ECS have been deleted.
- Cloud-Init has been installed and configured on the ECS.
- All EVS data disks attached to the Linux ECS have been detached.

Through the Management Console

1. Log in to the management console.
2. Under **Computing**, click **Elastic Cloud Server**.

   The **Elastic Cloud Server** page is displayed.
3. On the **Elastic Cloud Server** page, select the target ECS and ensure that the ECS is in the **Stopped** state.

   If the ECS is in the **running** state, click **More** in the **Operation** column and select **Stop** from the drop-down list to stop the ECS.
4. Click **More** in the **Operation** column and select **Make Image** from the drop-down list.
5. Enter basic image information as prompted.
   - **Source**: Select **ECS**.
   - **ECS**: Retain the default value.
   - **Name**: Customize your image name.
6. Click **Apply Now**.

   You will be redirected to the IMS console, on which you can view the created private image.

API

- URI

  POST /v2/cloudimages/action
- Example request

```
POST /v2/cloudimages/action
{
    "name": "ims_test",
    "description":"Create an image using an ECS.",
    "instance_id": "877a2cda-ba63-4e1e-b95f-e67e48b6129a"
}
```

- Example response

```
STATUS CODE 200
{
    "job_id": "8a12fc664fb4daa3014fb4e581380005"
}
```

2.5 Creating an Application Cluster

Scenarios

You can create multiple ECSs in batches within several minutes. This section describes how to create an application cluster using a private image on the management console or by calling HTTPS-based APIs.

Through the Management Console

1. Log in to the management console.
2. Under **Computing**, click **Elastic Cloud Server**.

   The **Elastic Cloud Server** page is displayed.

3. Click **Create ECS**.
4. Set ECS parameters as prompted. For details, see section **Creating an ECS That Supports InfiniBand NICs**.
   - **Specifications**: must be consistent with those of the ECS used to create the private image.
   - **Image**: Select **Private image** and then the private image created in step **Creating a Private Image Using an ECS**.
   - **VPC**: Select the VPC to which all ECSs in the HPC cluster belong.
   - **EIP**: Select **Not required**.

     NOTE

     Each cluster requires only one EIP. Therefore, bind an EIP after creating an application cluster.
   - **Quantity**: specifies the number of ECSs to be created.
5. Click **Next**.
6. On the page for you to confirm ECS configurations, view details about the ECSs.

   After confirming ECS configurations, click **Submit**.

   After the ECSs are created, you can view details about them on the ECS list. These ECSs function as an HPC cluster.

Through APIs

The following operations describe how to create an H2 ECS cluster:

- URI

  POST /v1/{$tenant_id}/cloudservers
- Example request

  For example, if you want to create four ECSs, change the value of **count** to **4**. An example request is as follows:

```
curl -i -k -H 'Accept:application/json;charset=utf8' -H 'Content-Type:application/json' -H 'X-Auth-
Token:$TOKEN' -d '{"server":{"availability_zone":"eu-
de-01","name":"h2_cluster_vm","imageRef":"7474de73-9618-4c6a-afaa-df60df57c9b9","flavorRef":"
h2.3xlarge.10","root_volume":{"volumetype":"SATA","size":40},"vpcid":"97701dc4-
bfd3-4021-8b89-044486c8b317","nics":[{"subnet_id":"6712fc43-
a196-4973-8b5e-5e4763f6449b"}],"personality":[],"count":4,"adminPass":"Test@123"}}' -X POST
https://46.29.103.37:443/v1/240bb6c5e42849669fc49933c185232b/cloudserver
```

  NOTE

  If each ECS in the cluster must have an EIP bound to it, multiple EIPs must be created. For details, see the "API" part in section **Creating an ECS That Supports InfiniBand NICs**.

2.6 Configuring Password-free Login Between ECSs in a Cluster

Scenarios

This section describes how to configure an ECS cluster to enable password-free login between ECSs in the cluster. Only password-authenticated ECSs support password-free logins in a cluster.

Background Information

$: indicates performing an operation as an administrator.

#: indicates performing an operation as an administrator.

Run the **sudo su** command to switch from a common user to an administrator.

Prerequisites

An ECS cluster has been created and an EIP has been bound to the cluster.

Procedure

1. Use PuTTY and a key pair to log in to an ECS in the cluster.
2. Run the following command to disable user logout upon system timeout:

   # TMOUT=0
3. Run the following command to add the private network IP addresses and names of all hosts in the cluster:

   # vi /etc/hosts

   Add the private network IP addresses and names of all ECSs in the cluster, for example:

   192.168.0.1 ecs-ff-0001

   192.168.0.2 ecs-ff-0002

   ..
4. Run the following commands to log in to the node using SSH and check whether you can log in to the ECS without a password (*hostname1* is the hostname):

   $ ssh localhost

   $ ssh *hostname1*
5. Log in to other ECSs in the cluster and repeat steps **1** to **4**.
6. Run the following commands to check whether the tested ECSs can be logged in to from each other without a password:

   For example, if the cluster contains two ECSs and the hostname of the other ECS is **hostname2**, run the following commands:

   $ ssh *Username@SERVER_IP*

   $ ssh hostname2

2.7 Running MPI Applications on an HPC Cluster

2.7.1 Open MPI Delivered with the IB Driver

Scenarios

This section describes how to run the built-in MPI (version 3.0.0rc6) of the IB driver on a configured ECS.

Prerequisites

- An ECS equipped with InfiniBand NICs has been created, and an EIP has been bound to it.
- Multiple ECSs have been created using a private image.

Procedure

1. Use PuTTY and a key pair to log in to the ECS.

   Ensure that the username specified during ECS creation is used to establish the connection.
2. Run the following command to disable user logout upon system timeout:

   # TMOUT=0
3. Run the following command to check whether the ECSs to be tested can be logged in to from each other without a password:

   $ ssh *Username*@*SERVER_IP*
4. Run the following commands to disable the firewall of the ECS:

   # iptables -F

   # service firewalld stop
5. Run the following command to log out of the **root** account:

   # exit
6. Run the following command to add the **hostfile** file:

   # vi hostfile

   Add the IP addresses or names of the ECSs (IP addresses corresponding with the names of ECSs are contained in the **/etc/hosts** directory). For example, add the following IP addresses:

   # cat hostfile

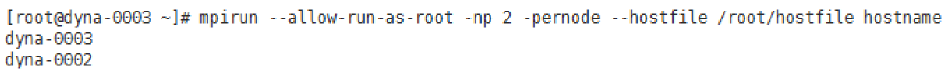
   192.168.0.1

   192.168.0.2

   ...
7. Run the following command to run the **hostname** command in the cluster:

   # mpirun --allow-run-as-root -np <hostfile_node_number> -pernode --hostfile hostfile hostname

Figure 2-29 Running the **hostname** command in the cluster



```
[root@dyna-0003 ~]# mpirun --allow-run-as-root -np 2 -pernode --hostfile /root/hostfile hostname
dyna-0003
dyna-0002
```

8. Modify **hostfile** and run MPI benchmark with the path of **hostfile** specified.

   For example, to modify the **hostfile** file and run MPI benchmark on two ECSs, run the following command:

   # mpirun --allow-run-as-root -np 2 -pernode --hostfile /root/hostfile /usr/mpi/gcc/openmpi-3.0.0rc6/tests/imb/IMB-MPI1 PingPong

   Run Intel MPI benchmark in a cluster containing two nodes. In the RDMA network, the minimum latency is less than 1.5 us.

```
#------------------------------------------------------------
#    Intel (R) MPI Benchmarks 4.1, MPI-1 part
#------------------------------------------------------------
# Date                  : Mon Jul 16 10:12:51 2018
# Machine               : x86_64
# System                : Linux
# Release               : 3.10.0-514.10.2.el7.x86_64
# Version               : #1 SMP Fri Mar 3 00:04:05 UTC 2017
# MPI Version           : 3.1
# MPI Thread Environment:

# New default behavior from Version 3.2 on:

# the number of iterations per message size is cut down
# dynamically when a certain run time (per message size sample)
# is expected to be exceeded. Time limit is defined by variable
# "SECS_PER_SAMPLE" (=> IMB_settings.h)
# or through the flag => -time

# Calling sequence was:

# /usr/mpi/gcc/openmpi-3.0.0rc6/tests/imb/IMB-MPI1 PingPong

# Minimum message length in bytes:   0
# Maximum message length in bytes:   4194304
#
# MPI_Datatype                   :   MPI_BYTE
# MPI_Datatype for reductions    :   MPI_FLOAT
# MPI_Op                         :   MPI_SUM
#
#

# List of Benchmarks to run:

# PingPong

#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
#bytes #repetitions      t[usec]   Mbytes/sec
0         1000         1.87         0.00
1         1000         1.93         0.49
2         1000         1.78         1.07
4         1000         1.79         2.13
8         1000         1.77         4.31
16         1000         1.78         8.57
32         1000         1.79         17.09
64         1000         1.85         33.02
128         1000         1.90         64.12
256         1000         2.40         101.58
512         1000         2.53         192.90
1024         1000         2.85         342.61
2048         1000         3.23         604.14
```

```
4096         1000         4.32       904.98
8192         1000         5.89      1325.65
16384         1000         8.48      1842.47
32768         1000        12.50      2500.57
65536          640        21.79      2867.89
131072          320        34.28      3646.50
262144          160        42.19      5925.52
524288           80        66.55      7513.14
1048576           40       114.95      8699.54
2097152           20       213.71      9358.48
4194304           10       402.59      9935.78


# All processes entering MPI_Finalize
```

9. Deploy your MPI application in the Linux cluster and run the MPI application in the Linux cluster using the preceding method.

2.7.2 Community Open MPI

Scenarios

This section describes how to run community Open MPI (version 3.1.1) on a configured ECS.

Prerequisites

- An ECS equipped with InfiniBand NICs has been created, and an EIP has been bound to it.
- Multiple ECSs have been created using a private image.

Procedure

1. Use PuTTY and a key pair to log in to the ECS.

   Ensure that the username specified during ECS creation is used to establish the connection.
2. Run the following command to disable user logout upon system timeout:

   # TMOUT=0
3. Run the following command to check whether the ECSs to be tested can be logged in to from each other without a password:

   $ ssh *Username*@*SERVER_IP*
4. Run the following commands to disable the firewall of the ECS:

   # iptables -F

   # service firewalld stop
5. Run the following command to set the hostnames of tested ECSs:

   # hostnamectl set-hostname vm1
6. Run the following command to add the **/etc/hosts** file:

   # vi /etc/hosts

   Add the hostnames and IP addresses of ECSs. The following are examples:

   #cat /etc/hosts

   192.168.1.3 vm1

   192.168.1.4 vm2

...

7. Run the following command to add the **hostfile** file:

   # vi hostfile

   Add the hostnames of the tested ECSs. The following are examples:

   vm1

   vm2

   ...

8. Modify **hostfile** and run MPI benchmark with the path of **hostfile** specified.

   For example, to modify the **hostfile** file and run MPI benchmark on two ECSs, run the following command:

   # mpirun --allow-run-as-root -np 2 --pernode -hostfile /root/hostfile /usr/mpi/gcc/openmpi-3.0.0rc6/tests/imb/IMB-MPI1 PingPong

   Run Intel MPI benchmark in a cluster containing two nodes. In the RDMA network, the minimum latency is less than 1.5 us.

```
#------------------------------------------------------------
#    Intel (R) MPI Benchmarks 4.1, MPI-1 part
#------------------------------------------------------------
# Date                  : Mon Jul 16 09:42:15 2018
# Machine               : x86_64
# System                : Linux
# Release               : 3.10.0-514.10.2.el7.x86_64
# Version               : #1 SMP Fri Mar 3 00:04:05 UTC 2017
# MPI Version           : 3.1
# MPI Thread Environment:

# New default behavior from Version 3.2 on:

# the number of iterations per message size is cut down
# dynamically when a certain run time (per message size sample)
# is expected to be exceeded. Time limit is defined by variable
# "SECS_PER_SAMPLE" (=> IMB_settings.h)
# or through the flag => -time

# Calling sequence was:

# /usr/mpi/gcc/openmpi-3.0.0rc6/tests/imb/IMB-MPI1 PingPong

# Minimum message length in bytes:   0
# Maximum message length in bytes:   4194304
#
# MPI_Datatype                   :   MPI_BYTE
# MPI_Datatype for reductions    :   MPI_FLOAT
# MPI_Op                         :   MPI_SUM
#
#

# List of Benchmarks to run:

# PingPong

#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
#bytes #repetitions      t[usec]   Mbytes/sec
0         1000         1.75         0.00
1         1000         1.75         0.55
2         1000         1.74         1.10
4         1000         1.74         2.19
8         1000         1.77         4.31
16         1000         1.79         8.54
```

```
32        1000       1.77       17.26
64        1000       1.85       33.02
128        1000       1.89       64.45
256        1000       2.39      102.29
512        1000       2.54      192.56
1024        1000       2.81      346.99
2048        1000       3.24      603.08
4096        1000       4.30      907.66
8192        1000       5.91     1321.23
16384        1000       8.61     1814.29
32768        1000      12.31     2537.83
65536         640      21.80     2867.15
131072         320      33.91     3686.23
262144         160      42.65     5861.95
524288          80      68.61     7287.12
1048576          40     120.06     8329.50
2097152          20     221.55     9027.12
4194304          10     424.35     9426.16


# All processes entering MPI_Finalize
```

9. Deploy your MPI application in the Linux cluster and run the MPI application in the Linux cluster using the preceding method.

2.7.3 Spectrum MPI

Scenarios

This section describes how to run Spectrum MPI (IBM Spectrum MPI v10.1) on a configured ECS.

Prerequisites

- An ECS equipped with InfiniBand NICs has been created, and an EIP has been bound to it.
- Multiple ECSs have been created using a private image.

Procedure

1. Use PuTTY and a key pair to log in to the ECS.

   Ensure that the username specified during ECS creation is used to establish the connection.
2. Run the following command to disable user logout upon system timeout:

   # TMOUT=0
3. Run the following command to check whether the ECSs to be tested can be logged in to from each other without a password:

   $ ssh *Username*@*SERVER_IP*
4. Run the following commands to disable the firewall of the ECS:

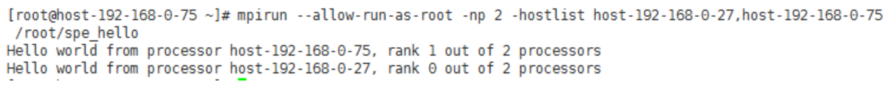
   # iptables -F

   # service firewalld stop
5. Run the executable file in the cluster using Spectrum MPI (use **IP:Number** as parameter **hostlist**).
   – **IP** indicates the IP addresses of the ECSs in the cluster.
   – **Number** indicates the number of tasks on an ECS.

For example, there are two ECSs with hostnames **host-192-168-0-27** and **host-192-168-0-75** in the cluster, and the executable file **spe_hello** is stored in **/root/spe_hello**. Then, run the following command:

# mpirun --allow-run-as-root -np 2 -hostlist host-192-168-0-27,host-192-168-0-75 /root/spe_hello

Figure 2-30 Running the executable file in the cluster using Spectrum MPI



```
[root@host-192-168-0-75 ~]# mpirun --allow-run-as-root -np 2 -hostlist host-192-168-0-27,host-192-168-0-75
 /root/spe_hello
Hello world from processor host-192-168-0-75, rank 1 out of 2 processors
Hello world from processor host-192-168-0-27, rank 0 out of 2 processors
```

NOTE

Specify the path of **hostfile** when running it. The path of the executable file **hello** must be absolute. All executable files in the cluster must be in the same directory.

2.7.4 Intel MPI

Scenarios

This section describes how to run Intel MPI (version l_mpi_2018.0.128) in an ECS cluster (take CentOS 7.3 as an example).

Prerequisites

- An ECS equipped with InfiniBand NICs has been created, and an EIP has been bound to it.
- Multiple ECSs have been created using a private image.

Procedure

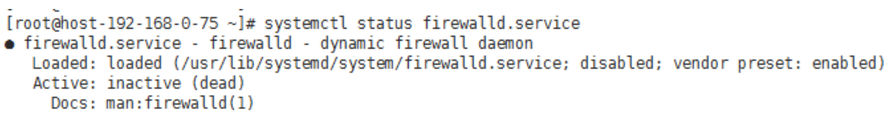
Step 1 Disable the firewall.

1. Log in to an ECS in the cluster.
2. Run the following command to disable the ECS firewall:

   # systemctl stop firewalld.service
3. Run the following command to check whether the firewall has been disabled:

   # systemctl status firewalld.service

   Figure 2-31 Disabling a firewall

```
[root@host-192-168-0-75 ~]# systemctl status firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

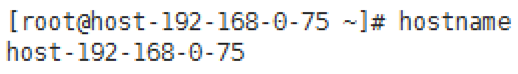
4. Log in to all other ECSs in the cluster and repeat **Step 1.1** to **Step 1.3** to disable firewalls on all ECSs.

Step 2 Modify the configuration file.

1. Log in to an ECS in the cluster.
2. Run the following command to view the ECS hostname:

   # hostname

Figure 2-32 Querying the ECS hostname



```
[root@host-192-168-0-75 ~]# hostname
host-192-168-0-75
```

3. Log in to all other ECSs in the cluster and repeat **Step 2.1** to **Step 2.2** to obtain hostnames of all ECSs.
4. Log in to an ECS in the cluster.
5. Run the following command to add the hosts configuration file:

   # vi /etc/hosts

   Add the private IP addresses and hostnames of all ECSs in the cluster. For example, run the following commands:

   192.168.0.1 host-192-168-0-1

   192.168.0.2 host-192-168-0-2

   ...
6. Run the following command to add the **hostfile** file:

   # vi hostfile

   Add hostnames of all ECSs in the cluster. For example, run the following commands:

   host-192-168-0-1

   host-192-168-0-2

   ...
7. Log in to other ECSs in the cluster and repeat **Step 2.4** to **Step 2.6**.

Step 3 Configure the IP addresses of InfiniBand NICs.

1. Run the following commands on all ECSs in the cluster to configure IP addresses for IB drivers:

   # ifconfig ib0 ***192.168.23.34*****/24**

   # ifconfig ib0 ***192.168.23.35*****/24**

   ...

   NOTE

   You can specify the IP addresses randomly, but ensure that they are in the same network segment.
2. Run the following command on an ECS to check network connectivity:

   # ping ***192.168.23.35***

Step 4 Run the following command to run Intel MPI on the ECS cluster:

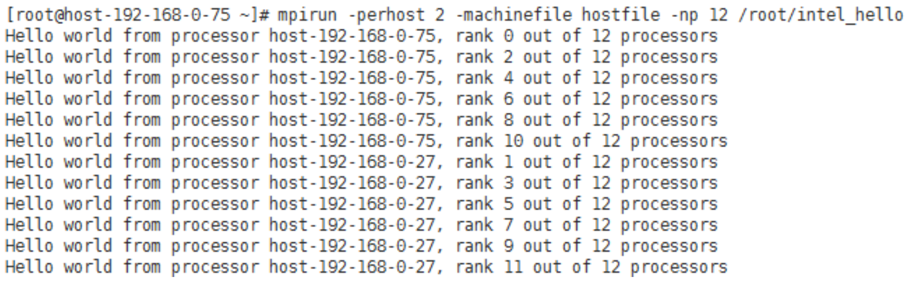
For example, there are two ECSs in the cluster. Then, run the following command:

# mpirun -perhost 2 -machinefile hostfile -np 12 /root/intel_hello

NOTE

Specify the path of **hostfile** when running it. The path of the executable file **hello** must be absolute. All executable files in the cluster must be in the same directory.

Figure 2-33 Running Intel MPI on the ECS cluster



```
[root@host-192-168-0-75 ~]# mpirun -perhost 2 -machinefile hostfile -np 12 /root/intel_hello
Hello world from processor host-192-168-0-75, rank 0 out of 12 processors
Hello world from processor host-192-168-0-75, rank 2 out of 12 processors
Hello world from processor host-192-168-0-75, rank 4 out of 12 processors
Hello world from processor host-192-168-0-75, rank 6 out of 12 processors
Hello world from processor host-192-168-0-75, rank 8 out of 12 processors
Hello world from processor host-192-168-0-75, rank 10 out of 12 processors
Hello world from processor host-192-168-0-27, rank 1 out of 12 processors
Hello world from processor host-192-168-0-27, rank 3 out of 12 processors
Hello world from processor host-192-168-0-27, rank 5 out of 12 processors
Hello world from processor host-192-168-0-27, rank 7 out of 12 processors
Hello world from processor host-192-168-0-27, rank 9 out of 12 processors
Hello world from processor host-192-168-0-27, rank 11 out of 12 processors
```

----End

2.7.5 Platform MPI

Scenarios

This section describes how to run Platform MPI (version platform_mpi-09.01.04.03r-ce) in an ECS cluster (take CentOS 7.3 as an example).

Prerequisites

- An ECS equipped with InfiniBand NICs has been created, and an EIP has been bound to it.
- Multiple ECSs have been created using a private image.

Procedure

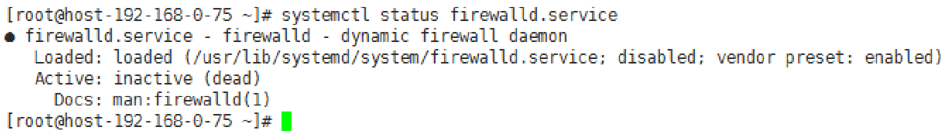
Step 1 Disable the firewall.

1. Log in to an ECS in the cluster.
2. Run the following command to disable the ECS firewall:

   # systemctl stop firewalld.service
3. Run the following command to check whether the firewall has been disabled:

   # systemctl status firewalld.service

   Figure 2-34 Disabled firewall

```
[root@host-192-168-0-75 ~]# systemctl status firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
[root@host-192-168-0-75 ~]#
```

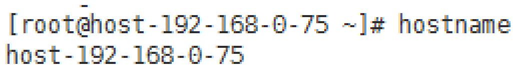
4. Log in to all other ECSs in the cluster and repeat **Step 1.1** to **Step 1.3** to disable firewalls on all ECSs.

Step 2 Modify the configuration file.

1. Log in to an ECS in the cluster.
2. Run the following command to view the ECS hostname:

   # hostname

   Figure 2-35 Viewing the ECS hostname

```
[root@host-192-168-0-75 ~]# hostname
host-192-168-0-75
```

3. Log in to all other ECSs in the cluster and repeat **Step 2.1** to **Step 2.2** to obtain hostnames of all ECSs.
4. Log in to an ECS in the cluster.
5. Run the following command to add the hosts configuration file:

   # vi /etc/hosts

   Add the private IP addresses and hostnames of all ECSs in the cluster. For example, run the following commands:

   192.168.0.1 host-192-168-0-1

   192.168.0.2 host-192-168-0-2

   ...
6. Run the following command to add the **hostfile** file:

   $vi hostfile

   Add hostnames of all ECSs in the cluster. For example, run the following commands:

   host-192-168-0-1

   host-192-168-0-1

   ...
7. Log in to all ECSs in the cluster and repeat **Step 2.4** to **Step 2.6**.

Step 3 Configure the IP addresses of InfiniBand NICs.

1. Run the following commands on all ECSs in the cluster to configure IP addresses for IB drivers:

   # ifconfig ib0 *192.168.23.34***/24**

   # ifconfig ib0 *192.168.23.35***/24**

   ...

   NOTE

   You can specify the IP addresses randomly, but ensure that they are in the same network segment.
2. Run the following command on an ECS to check network connectivity:

   # ping *192.168.23.35*

Step 4 Run the following command to run Platform MPI in the ECS cluster:

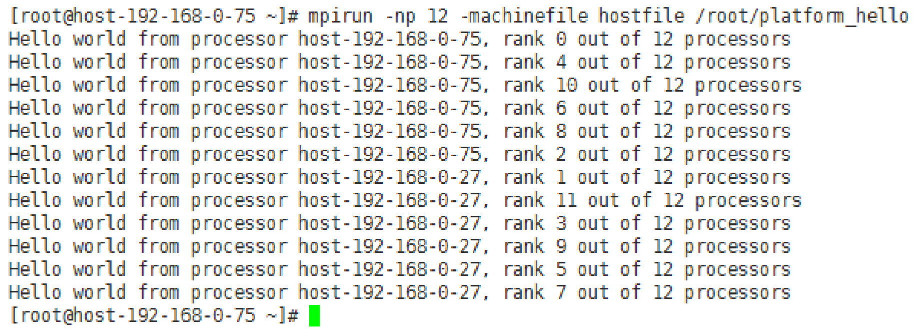
For example, there are two ECSs in the cluster. Then, run the following command:

# mpirun -perhost 2 -np 12 -machinefile hostfile /root/platform_hello

NOTE

Specify the path of **hostfile** when running it. The path of the executable file **hello** must be absolute. All executable files in the cluster must be in the same directory.

Figure 2-36 Successful execution of Platform MPI in the ECS cluster


```
[root@host-192-168-0-75 ~]# mpirun -np 12 -machinefile hostfile /root/platform_hello
Hello world from processor host-192-168-0-75, rank 0 out of 12 processors
Hello world from processor host-192-168-0-75, rank 4 out of 12 processors
Hello world from processor host-192-168-0-75, rank 10 out of 12 processors
Hello world from processor host-192-168-0-75, rank 6 out of 12 processors
Hello world from processor host-192-168-0-75, rank 8 out of 12 processors
Hello world from processor host-192-168-0-75, rank 2 out of 12 processors
Hello world from processor host-192-168-0-27, rank 1 out of 12 processors
Hello world from processor host-192-168-0-27, rank 11 out of 12 processors
Hello world from processor host-192-168-0-27, rank 3 out of 12 processors
Hello world from processor host-192-168-0-27, rank 9 out of 12 processors
Hello world from processor host-192-168-0-27, rank 5 out of 12 processors
Hello world from processor host-192-168-0-27, rank 7 out of 12 processors
[root@host-192-168-0-75 ~]#
```

----End

3 Best Practices in the ECS Scenario

3.1 HPC Resumable Computing Solution

Scenarios

Many HPC applications support resumable computing, such as LAMMPS and GROMACS. In addition, common HPC scheduling software can have resumable computing integrated, such as PBS, Slurm, and LSF.

This section uses LAMMPS as an example to describe how to perform HPC resumable computing.

Step 1: Install FFTW

Run the following commands to install FFTW:

yum install gcc-gfortran gcc-c++

wget http://www.fftw.org/fftw-3.3.8.tar.gz

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/mpi/gcc/*openmpi-2.1.2a1*/lib64/

export PATH=/usr/mpi/gcc/*openmpi-2.1.2a1*/bin:$PATH

tar -zxvf fftw-3.3.8.tar.gz

cd fftw-3.3.8/

./configure --prefix=/opt/fftw CC=gcc MPICC=mpicc --enable-mpi --enable-openmp --enable-threads --enable-avx --enable-shared

make && make install

Step 2: Install LAMMPS

1. Run the following commands to install LAMMPS:
   yum install libjpeg-*
   yum install libpng12-*

wget https://lammps.sandia.gov/tars/lammps-2Aug18.tar.gz

tar -zxvf lammps-2Aug18.tar.gz

cd lammps-2Aug18/src

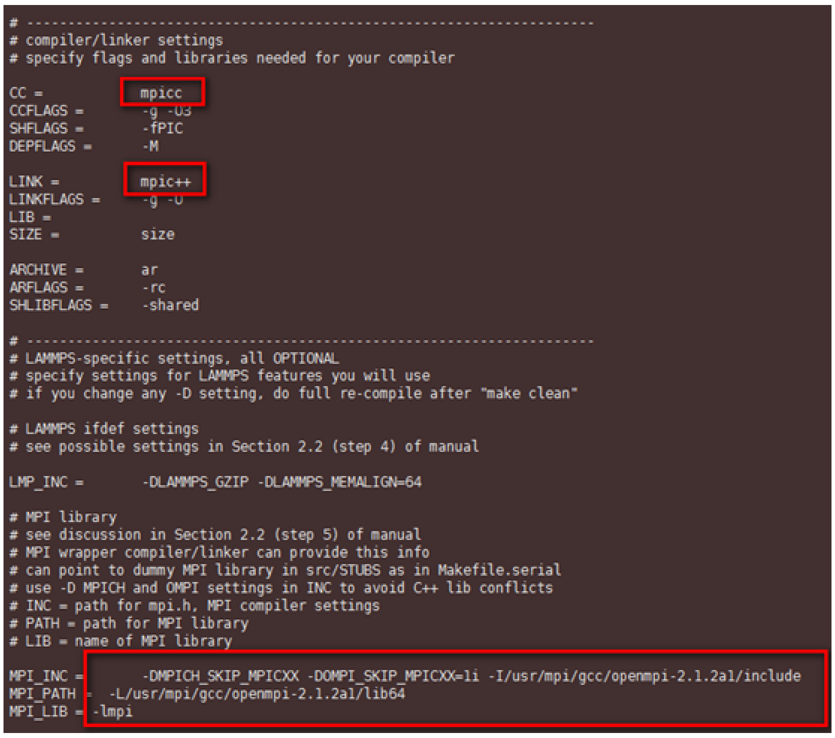
vi MAKE/Makefile.mpi

2. Modify the data marked in red boxes in **Figure 3-1** and **Figure 3-2**. Change the version based on site requirements.

NOTICE

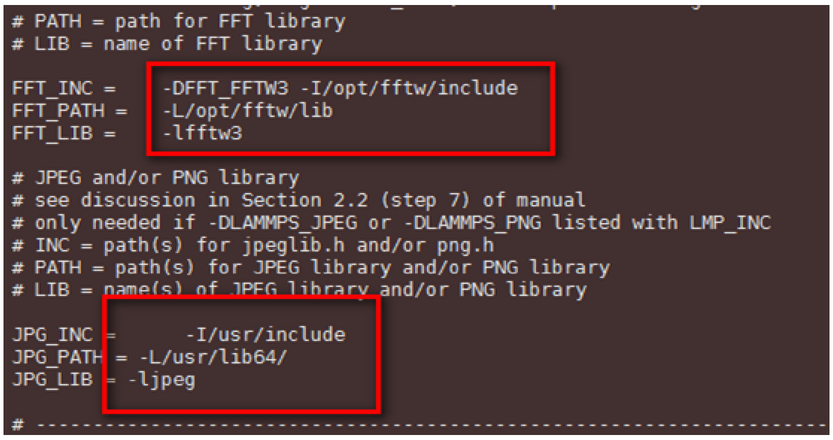
Modify only the data marked in red boxes in **Figure 3-1** and **Figure 3-2**.

Figure 3-1 Modifying the **Makefile** file 1



```
# ---------------------------------------------------------------------
# compiler/linker settings
# specify flags and libraries needed for your compiler

CC =            mpicc
CCFLAGS =       -g -O3
SHFLAGS =       -fPIC
DEPFLAGS =      -M

LINK =          mpic++
LINKFLAGS =     -g -O
LIB =
SIZE =          size

ARCHIVE =       ar
ARFLAGS =       -rc
SHLIBFLAGS =    -shared

# ---------------------------------------------------------------------
# LAMMPS-specific settings, all OPTIONAL
# specify settings for LAMMPS features you will use
# if you change any -D setting, do full re-compile after "make clean"

# LAMMPS ifdef settings
# see possible settings in Section 2.2 (step 4) of manual

LMP_INC =       -DLAMMPS_GZIP -DLAMMPS_MEMALIGN=64

# MPI library
# see discussion in Section 2.2 (step 5) of manual
# MPI wrapper compiler/linker can provide this info
# can point to dummy MPI library in src/STUBS as in Makefile.serial
# use -D MPICH and OMPI settings in INC to avoid C++ lib conflicts
# INC = path for mpi.h, MPI compiler settings
# PATH = path for MPI library
# LIB = name of MPI library

MPI_INC =       -DMPICH_SKIP_MPICXX -DOMPI_SKIP_MPICXX=1i -I/usr/mpi/gcc/openmpi-2.1.2a1/include
MPI_PATH =  -L/usr/mpi/gcc/openmpi-2.1.2a1/lib64
MPI_LIB = -lmpi
```

Figure 3-2 Modifying the **Makefile** file 2



```
# PATH = path for FFT library
# LIB = name of FFT library

FFT_INC =    -DFFT_FFTW3 -I/opt/fftw/include
FFT_PATH =   -L/opt/fftw/lib
FFT_LIB =    -lfftw3

# JPEG and/or PNG library
# see discussion in Section 2.2 (step 7) of manual
# only needed if -DLAMMPS_JPEG or -DLAMMPS_PNG listed with LMP_INC
# INC = path(s) for jpeglib.h and/or png.h
# PATH = path(s) for JPEG library and/or PNG library
# LIB = name(s) of JPEG library and/or PNG library

JPG_INC =       -I/usr/include
JPG_PATH = -L/usr/lib64/
JPG_LIB = -ljpeg

# ---------------------------------------------------------------------
```

3. Run the following command to compile LAMMPS and copy the obtained **lmp_mpi** file to **/share**:

   make mpi

Step 3: Configure LAMMPS

1. Configure the example input file.

   Melt is used as an example to generate example file **melt.in**. For example, a checkpoint file is automatically generated for every 100 iterative operations, and the file is stored in **/share**. The file is as follows:

```
# 3d Lennard-Jones melt

units           lj
atom_style      atomic

lattice         fcc 0.8442
region          box block 0 20 0 20 0 20
create_box      1 box
create_atoms    1 box
mass            1 1.0

velocity        all create 1.44 87287 loop geom

pair_style      lj/cut 2.5
pair_coeff      1 1 1.0 1.0 2.5

neighbor        0.3 bin
neigh_modify    delay 5 every 1

fix             1 all nve
dump 1 all xyz 100 /share/sample.xyz
run             10000 every 100 "write_restart /share/lennard.restart"
```

2. Obtain the **melt.restart.in** input file for resumable checkpoint computing.

```
# 3d Lennard-Jones melt

read_restart  /share/lennard.restart
run         10000 every 100 "write_restart /share/lennard.restart"
```

3. Obtain the PBS job script **job.pbs**.

```
#!/bin/sh
#PBS -l ncpus=2
#PBS -o lammps_pbs.log
#PBS -j oe

export PATH=/usr/mpi/gcc/openmpi-2.1.2a1/bin:$PATH

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/mpi/gcc/openmpi-2.1.2a1/lib64/module

if [ ! -e "/share/lennard.restart" ]; then
  echo "run at the beginning"
  mpiexec --allow-run-as-root -np 2 /share/lmp_mpi -in /share/melt.in
else
  echo "run from the last checkpoint"
  mpiexec --allow-run-as-root -np 2 /share/lmp_mpi -in /share/melt.restart.in
fi
```

Step 4: Submit a Job and Ensure that the Job Is Not Interrupted During Running

Submit and run the job without interrupting it, and check the job running duration.

1. Run the following command to submit a job:

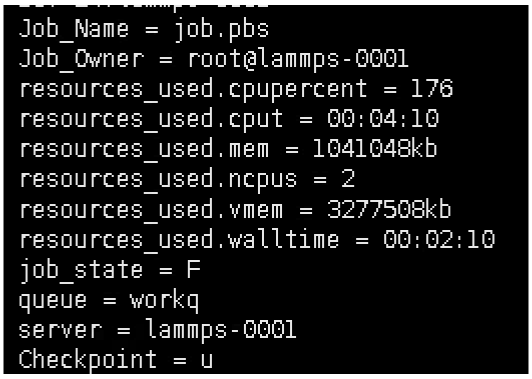
   qsub job.pbs

2. After the job is complete, run the following command to view the job information:

   qstat -f *Job ID*

   As shown in **Figure 3-3**, the job runs for 4 minutes and 10 seconds.

Figure 3-3 Job running without being interrupted



```
Job_Name = job.pbs
Job_Owner = root@lammps-0001
resources_used.cpupercent = 176
resources_used.cput = 00:04:10
resources_used.mem = 1041048kb
resources_used.ncpus = 2
resources_used.vmem = 3277508kb
resources_used.walltime = 00:02:10
job_state = F
queue = workq
server = lammps-0001
Checkpoint = u
```

Step 5: Submit a Job, Emulate a Computing Interrupt, and Use Resumable Computing to Complete the Computing

After submitting a job, stop the compute node to emulate a computing interruption. Then, check the job running durations before and after the interruption.

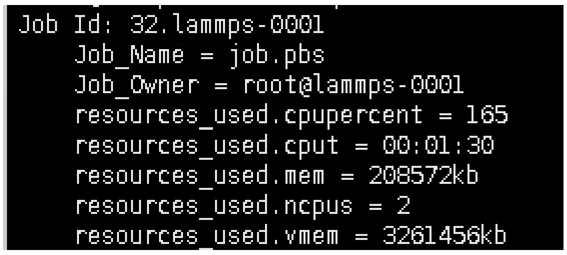
1. Run the following command to submit a job:

   qsub job.pbs

2. After the job runs for about 1 minute and 30 seconds, stop the compute node on which the job runs to emulate example release.
3. Run the following command to check the job information after the compute node is stopped:

   qstat -f *Job ID*

Figure 3-4 Job running before interruption



```
Job Id: 32.lammps-0001
    Job_Name = job.pbs
    Job_Owner = root@lammps-0001
    resources_used.cpupercent = 165
    resources_used.cput = 00:01:30
    resources_used.mem = 208572kb
    resources_used.ncpus = 2
    resources_used.vmem = 3261456kb
```

   In such a case, the job returns back to the **queued** state, waiting for available computing resources.

4. Start the compute node to provide available computing resources.

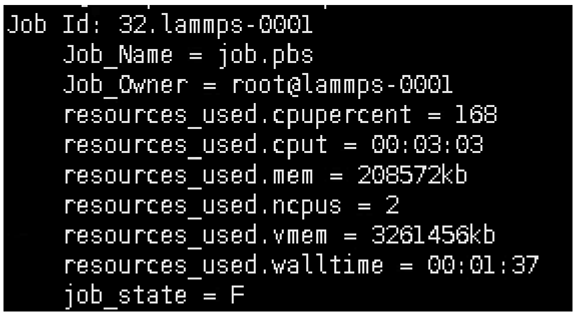
   In such a case, the job will continue.

5. After the job is complete, run the following command to view the job information:

   qstat -f *Job ID*

   As shown in **Figure 3-5**, the job runs for 3 minutes and 3 seconds. It is shown that the job computing is resumed at the time when the computing is interrupted.

   Figure 3-5 Job running after interruption



```
Job Id: 32.lammps-0001
    Job_Name = job.pbs
    Job_Owner = root@lammps-0001
    resources_used.cpupercent = 168
    resources_used.cput = 00:03:03
    resources_used.mem = 208572kb
    resources_used.ncpus = 2
    resources_used.vmem = 3261456kb
    resources_used.walltime = 00:01:37
    job_state = F
```

4 Typical Applications in the BMS Scenario

4.1 Creating a BMS Cluster

Scenarios

This section describes how to allocate a BMS so that you can deploy your services on it.

Precautions

Dedicated physical resources

- To make a BMS run on isolated physical hardware, apply for a Dedicated Cloud (DeC) and then create a BMS.

  For details about DeC and how to apply for DeC, see *Dedicated Cloud User Guide*.
- If you want your BMS to exclusively use a storage device, apply for Dedicated Enterprise Storage Service (DESS) after you enable DeC and then create the BMS.

  For details about DESS, see *Dedicated Enterprise Storage Service User Guide*.

Procedure

1. Log in to the management console.
2. Under **Computing**, click **Bare Metal Server**.

   The BMS console is displayed.
3. Click **Buy BMS**.

   The page for you to purchase a BMS is displayed.

   NOTE

   If you have obtained a DeC and want to deploy your BMS in your DeC, click **Provision BMS in DeC**.
4. Confirm **Region**.

If the region is incorrect, click in the upper left corner of the page for correction.

5. Set **AZ**.

   An AZ is a physical region where power and networks are physically isolated. AZs in the same region can communicate with each other over an intranet. It is recommended that you apply for BMSs in different AZs to ensure high availability of applications running on the BMSs.

6. Set **Flavor**.

   The flavor contains the CPU, memory, local disks, and extended configuration of the BMS. After you select a flavor, the name, networking, and use scenarios of the flavor are displayed under the flavor list.

   NOTE

   Configuration in the flavor, such as memory and local disks, cannot be changed.

7. Set **Image**.
   - Public image

     A public image is a standard, widely used image. It contains an OS and preinstalled public applications and is available to all users.

     To select a public image, set **Image** to **Public image** and select a desired one from the drop-down lists.
   - Private image

     A private image is an image available only to the user who created it using an external image. It contains an OS, SDI card driver, bms-network-config network configuration program, Cloud-Init, and users' private applications.

     To select a private image, set **Image** to **Private image** and select a desired one from the drop-down list.
   - Shared image

     A shared image is a private image shared by another user.

     To select a shared image, click **Shared image** and select a desired one from the drop-down list.

8. Select **License Type**.

   Set a license type for using an OS or software on the cloud platform. This parameter is available only if the image you selected is charged.
   - Use the system license

     Allows you to use the license provided by the cloud platform. Obtaining the authorization of such a license is charged.
   - Bring your own license (BYOL)

     Allows you to use your existing OS license. In such a case, you do not need to apply for a license again.

9. Set **Disk**.

   Disks are classified as EVS disks, DSS disks, and DESS disks based on whether the storage resources used by the disks are exclusive. Currently, you can only select one disk type.
   - **EVS**: provides secure, reliable, and scalable hard disk resources of various flavors to meet performance requirements in different scenarios.

If you have not applied for an exclusive storage pool, click **EVS** and create EVS disks that use public storage resources.

- **DSS**: provides exclusive storage resources, high availability and durability, and stable and low latency using multiple technologies, such as data redundancy and cache acceleration.

  If you have applied for a storage pool on the DSS page, click **DSS** and create disks in the obtained storage pool.

- **DESS**: provides dedicated storage devices for enterprises to migrate key services (such as Oracle RAC and SAP HANA TDI) to the cloud.

  If you have enabled the Dedicated Enterprise Storage Service (DESS), click **DESS** and create disks in DESS.

A disk can be a system or data disk. You can add multiple data disks to a BMS and customize the system disk size.

NOTE

Windows BMSs cannot have disks attached.

- System disk

  If you select a flavor that supports quick provisioning, the system disk is available. You can set the disk type and size as needed.

- Data disk

  You can add multiple data disks to a BMS and configure sharing for each data disk.

  - Currently, BMSs only support SCSI disks.
  - **Share**: indicates that the disk is shared. A shared disk can be attached to multiple BMSs.

NOTE

- After a system disk is detached from a BMS charged in yearly/monthly mode, the disk can only be used as a system disk and can only be attached to this BMS.
- If you detach a non-shared data disk purchased when you buy a BMS charged in yearly/monthly mode and want to attach it again, you can only attach it to the original BMS as a data disk.
- The non-shared data disk purchased when you buy a BMS charged in yearly/monthly mode does not support separate renewal, unsubscription, automatic service renewal, conversion to on-demand payment, and release.

10. Configure automatic backup.

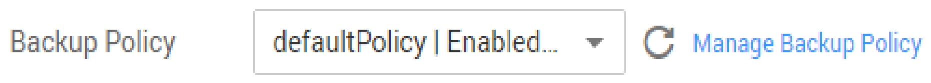
    After automatic backup is enabled, the system automatically backs up the BMS based on the preset backup policy.

    a. Select **Enable**.

    b. Configure **Backup Policy**.

       In the drop-down list, select a backup policy. Alternatively, you can click **Manage Backup Policy** and set the backup policy on the Cloud Server Backup Service (CSBS) page. If you have not created any backup policy but have selected **Enable**, the system will use the default backup policy, as shown in **Figure 4-1**.

Figure 4-1 Default backup policy



For more information about BMS backup, see *Cloud Server Backup Service User Guide*.

11. Set network parameters, including **VPC**, **Security Group**, and **NIC**.

Table 4-1 Network parameters

Parameter	Description
VPC	If you do not have a VPC configured when creating a BMS for the first time, the system creates the default VPC, subnet, and security group for you. The default subnet segment is 192.168.1.0/24 and the subnet gateway is 192.168.1.1. Dynamic Host Configuration Protocol (DHCP) is enabled for the subnet. If the default VPC cannot meet your network requirements, click **View VPC** to create a VPC.
Security Group	A security group implements access control for BMSs within a security group and between different security groups. You can define different access control rules for a security group, and these rules take effect for all BMSs added to this security group. When allocating a BMS, you can select the security group to which it belongs. You can select only one security group when applying for a BMS. **NOTE** Before initializing a BMS, ensure that security group rules in the outbound direction meet the following requirements: ● Protocol: TCP ● Port Range: 80 ● Remote End: 169.254.0.0/16 If you use the default outbound security group rule, the preceding requirements are met, and the BMS can be initialized. The default outbound security group rule is as follows: ● Protocol: Any ● Port Range: Any ● Remote End: 0.0.0.0/16
NIC	Specifies the BMS NIC. The system provides a primary NIC by default. You can specify the IP address to use for the primary NIC in your VPC. You can add extended NICs to the BMS.

Parameter	Description
High-Speed NIC	Specifies the NIC that is assigned an IP address from the high-speed network segment. A high-speed NIC provides higher bandwidth for a BMS. A BMS can have a maximum number of two high-speed NICs. You can configure the IP address to be used by the high-speed NIC. **NOTE** If you use a high-speed NIC, you cannot create BMSs in batches.
EIP	An elastic IP address (EIP) is a fixed (static) IP address that you have bound to a BMS. The elastic IP address enables resources in a VPC to provide services using a fixed IP address. You can select one of the following three options for **EIP** as required: ● **Not required**: BMSs cannot communicate with the Internet and are used only as a service deployed in the private network or used in the cluster. ● **Automatically assign**: The system automatically assigns an EIP that uses exclusive bandwidth to the BMS. The bandwidth size is configurable. ● **Use existing**: An existing elastic IP address is assigned to the BMS. **NOTE** If you use an existing EIP, you cannot create BMSs in batches.
Specifications	This parameter is mandatory when **EIP** is set to **Automatically assign**. ● **Dynamic BGP**: When changes occur on a network using dynamic BGP, routing protocols provide automatic, real-time optimization of network configurations, ensuring network stability and optimal user experience. ● **Static BGP**: When changes occur on a network using static BGP, carriers cannot adjust network configurations in real time to ensure optimal user experience.
Billed By	This parameter is mandatory when **EIP** is set to **Automatically assign**. **Bandwidth**: You specify a maximum bandwidth and pay for the amount of time you use the bandwidth.
Bandwidth	This parameter is mandatory when **EIP** is set to **Automatically assign**. Specifies the bandwidth size in Mbit/s.

12. Set the BMS login mode.

- Key pair

  A key pair is used for BMS login authentication. You can select an existing key pair, or click **Create Key Pair** and create a desired one.

  NOTE

  If you use an existing key pair, make sure that you have saved the key file locally. Otherwise, you will fail to log in to the BMS.

- Password

  The initial password is used for authentication. You can log in to the BMS using the username and its initial password. If the BMS runs Linux, you can use username **root** and its initial password to log in to the BMS. If the BMS runs Windows, you can use username **Administrator** and its initial password to log in to the BMS. The passwords must meet the requirements described in **Table 4-2**.

  NOTE

  You cannot select this login mode for Windows BMSs.

Table 4-2 Password requirements

Parameter	Requirements	Example Value
Password	● Consists of 8 to 26 characters. ● Must contain at least three of the following character types: – Uppercase letters – Lowercase letters – Digits – Special characters !@$%^-_=+[]{}:,./? ● Cannot contain the username or the username spelled backwards. ● Cannot contain more than two characters in the same sequence as they appear in the username. (This requirement applies only to Windows BMSs.)	Test12$@

13. (Optional) Configure **Advanced Settings**.

    To use functions listed in **Advanced Settings**, click **Configure now**. Otherwise, click **Do not configure**.

    User Data Injection enables the BMS to automatically inject user data when the BMS starts for the first time. After this function is enabled, the BMS automatically injects the user data upon its first startup.

14. Set **BMS Name**.

    If you want to buy multiple BMSs at a time, the system automatically sequences these BMSs by adding suffixes to them.

15. Set **Required Duration** and **Quantity**.
    - **Required Duration**: Set the service duration if you select the **Yearly/ Monthly** billing mode. The service duration ranges from one month to one year.

      NOTE

      BMSs charged in yearly/monthly mode cannot be deleted. They support only resource unsubscription. If you no longer need a BMS, you can unsubscribe from it using either of the following methods:
      - Locate the row that contains the BMS, click **More** in the **Operation** column, and select **Unsubscribe** from the drop-down list. On the **Unsubscribe** page, select a reason and click **Confirm**.
      - Choose **Billing Center** > **Orders** > **Unsubscriptions**. Locate the row that contains the BMS and click **Unsubscribe from Resource** in the **Operation** column.
    - **Quantity**: **1** to **24**

      NOTE

      - If the quota is sufficient, you can buy a maximum of 24 BMSs. If the quota is less than 24, you can buy a maximum of all available BMSs.
      - If you manually set an IP address when configuring **NIC** or **High-Speed NIC** or select **Use existing** when configuring **EIP**, you can create only one BMS at a time.
16. Click **Buy Now**.

    On the confirmation page, confirm the BMS information and click **Pay Now**.

    NOTE

    If you have any question about the price, click **Pricing details**.
17. Pay the fees as prompted and click **OK**.

    The BMS console is displayed.
18. After you pay the order, the system will create your requested BMSs.

    The BMS status changes to **Running** after about 30 minutes.

    NOTE

    If you select a flavor that supports quick provisioning, you can obtain a BMS in about five minutes.

4.2 Configuring Password-free Login Between BMSs in a Cluster

Scenarios

This section describes how to configure a BMS cluster to enable password-free login between BMSs in the cluster.

Background

$: indicates performing an operation as a common user.

#: indicates performing an operation as an administrator.

Run the **sudo su** command to switch from a common user to an administrator.

Prerequisites

A BMS has been created and an EIP has been bound to the BMS.

Procedure

1. Use PuTTY and a key pair to log in to any BMS in the cluster.
2. Run the following command to disable user logout upon system timeout:

   # TMOUT=0
3. Copy the BMS key file in .pem format to the **.ssh** directory and name it **id_rsa**.

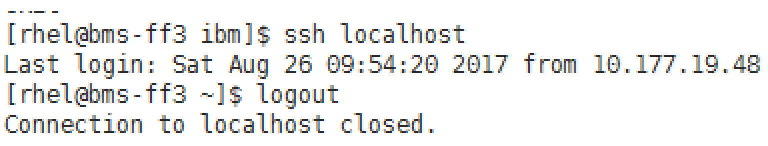
   $ cd ~/.ssh

   $ mv *.pem id_rsa
4. Run the following command to assign permissions of the key file:

   $ sudo chmod 600 id_rsa
5. Run the following command to log in to the BMS using SSH and check whether you can log in to the BMS without a password:

   $ ssh localhost

   Figure 4-2 Password-free login to a BMS

   ```
   [rhel@bms-ff3 ibm]$ ssh localhost
   Last login: Sat Aug 26 09:54:20 2017 from 10.177.19.48
   [rhel@bms-ff3 ~]$ logout
   Connection to localhost closed.
   ```

   If information shown in **Figure 4-2** is displayed and you can log in to the BMS without a password, the permissions have been assigned.
6. Log in to other BMSs in the cluster and repeat steps **1** to **5**.
7. Run the following commands to check whether the tested BMSs can be logged in from each other without a password:

   $ ssh *Username*@*SERVER_IP*

4.3 Installing and Using MPIs (x86 BMS)

This section uses the CentOS 7.3 OS as an example to describe how to run MPIs on a single node.

4.3.1 MPIs Supported in the BMS Scenario

The following MPIs are supported in the BMS scenario:

- Built-in OpenMPI of the IB driver
- Community OpenMPI
- Spectrum MPI

- Intel MPI
- Platform MPI

Install and use an MPI as needed.

4.3.2 Open MPI Delivered with the IB Driver

Scenarios

This section describes how to install and use Open MPI (version 3.1.0rc2 is used as an example) delivered with the IB driver on a BMS.

Perform the operations on each BMS in a cluster.

Prerequisites

Password-free login has been configured between BMSs in the cluster.

Procedure

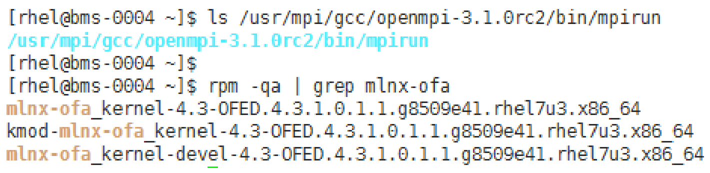
Step 1 Check whether the IB driver has been installed.

1. Run the following commands to check whether the IB driver has been installed:

   $ ls /usr/mpi/gcc/openmpi-3.1.0rc2/bin/mpirun

   $ rpm -qa | grep mlnx-ofa

   Figure 4-3 Installed IB driver

```
[rhel@bms-0004 ~]$ ls /usr/mpi/gcc/openmpi-3.1.0rc2/bin/mpirun
/usr/mpi/gcc/openmpi-3.1.0rc2/bin/mpirun
[rhel@bms-0004 ~]$
[rhel@bms-0004 ~]$ rpm -qa | grep mlnx-ofa
mlnx-ofa_kernel-4.3-OFED.4.3.1.0.1.1.g8509e41.rhel7u3.x86_64
kmod-mlnx-ofa_kernel-4.3-OFED.4.3.1.0.1.1.g8509e41.rhel7u3.x86_64
mlnx-ofa_kernel-devel-4.3-OFED.4.3.1.0.1.1.g8509e41.rhel7u3.x86_64
```

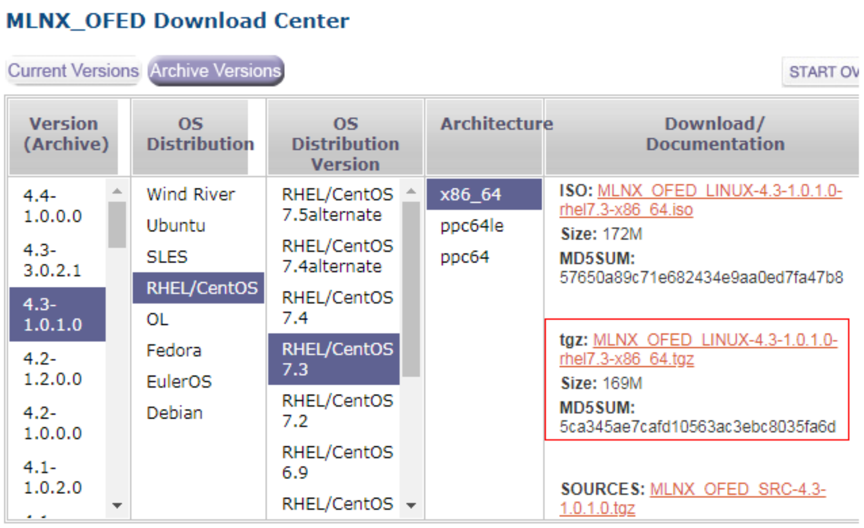
2. Check the command output.
   - If information shown in **Figure 4-3** is displayed, the IB driver has been installed. Then, go to **Step 3**.
   - If the IB driver has not been installed, go to **Step 2**.

Step 2 Install the IB driver.

1. Download the installation package **MLNX_OFED_LINUX-4.3-1.0.1.0-rhel7.3-x86_64.tgz**.

   Download path: **https://network.nvidia.com/products/infiniband-drivers/linux/mlnx_ofed/**

Figure 4-4 IB driver download center



ISO: MLNX_OFED_LINUX-4.3-1.0.1.0-rhel7.3-x86_64.iso
Size: 172M
MD5SUM: 57650a89c71e682434e9aa0ed7fa47b8

tgz: MLNX_OFED_LINUX-4.3-1.0.1.0-rhel7.3-x86_64.tgz
Size: 169M
MD5SUM: 5ca345ae7cafd10563ac3ebc8035fa6d

SOURCES: MLNX_OFED_SRC-4.3-1.0.1.0.tgz

2. Run the following commands to install the software package:

   # yum install tk tcl

   # tar -xvf MLNX_OFED_LINUX-4.3-1.0.1.0-rhel7.3-x86_64.tgz

   # cd MLNX_OFED_LINUX-4.3-1.0.1.0-rhel7.3-x86_64

   # ./mlnxofedinstall

Step 3 Configure environment variables.

1. Use VIM to open the **~/.bashrc** file and add the following data to the file:

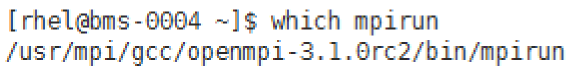
   export PATH=$PATH:/usr/mpi/gcc/openmpi-3.1.0rc2/bin

   export LD_LIBRARY_PATH=/usr/mpi/gcc/openmpi-3.1.0rc2/lib64

2. Run the following command to check whether the MPI environment variables are correct:

   $ which mpirun

   Figure 4-5 Viewing the Open MPI environment variables

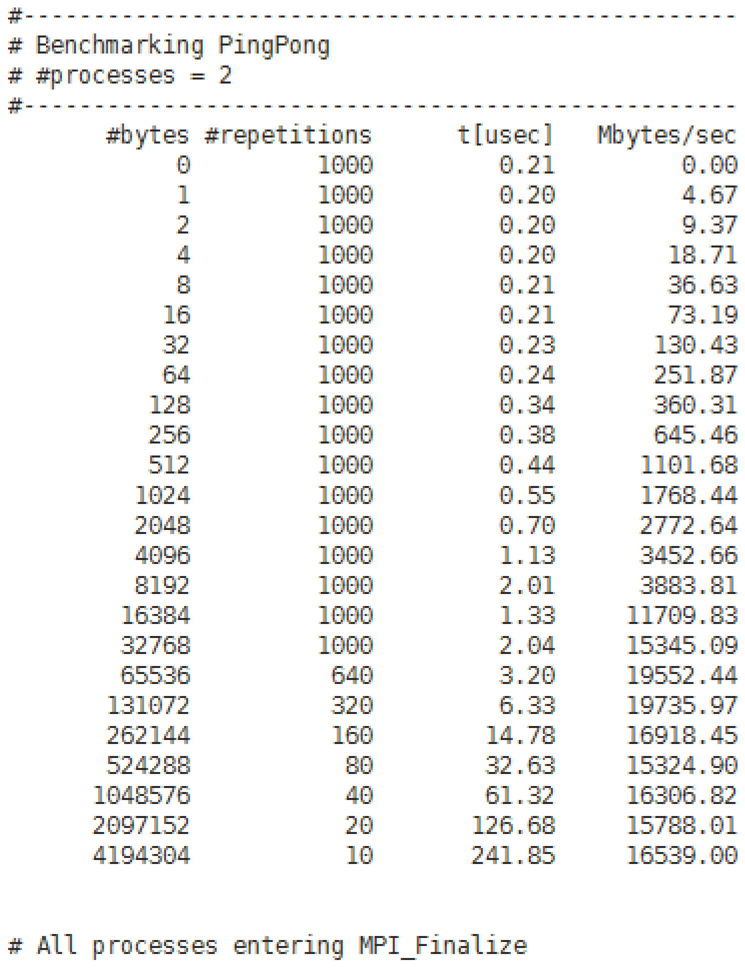
```
[rhel@bms-0004 ~]$ which mpirun
/usr/mpi/gcc/openmpi-3.1.0rc2/bin/mpirun
```

   If information shown in **Figure 4-5** is displayed, environment variables have been configured.

Step 4 Run the following command to run Open MPI delivered with the IB driver on a BMS:

$ mpirun -np 2 -mca btl_openib_if_include "mlx5_0:1" -x MXM_IB_USE_GRH=y /usr/mpi/gcc/openmpi-3.1.0rc2/tests/imb/IMB-MPI1 PingPong

Figure 4-6 Running the Open MPI on a BMS



```
#-----------------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#-----------------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec
            0         1000         0.21         0.00
            1         1000         0.20         4.67
            2         1000         0.20         9.37
            4         1000         0.20        18.71
            8         1000         0.21        36.63
           16         1000         0.21        73.19
           32         1000         0.23       130.43
           64         1000         0.24       251.87
          128         1000         0.34       360.31
          256         1000         0.38       645.46
          512         1000         0.44      1101.68
         1024         1000         0.55      1768.44
         2048         1000         0.70      2772.64
         4096         1000         1.13      3452.66
         8192         1000         2.01      3883.81
        16384         1000         1.33     11709.83
        32768         1000         2.04     15345.09
        65536          640         3.20     19552.44
       131072          320         6.33     19735.97
       262144          160        14.78     16918.45
       524288           80        32.63     15324.90
      1048576           40        61.32     16306.82
      2097152           20       126.68     15788.01
      4194304           10       241.85     16539.00


# All processes entering MPI_Finalize
```

----End

4.3.3 Community Open MPI

Scenarios

This section describes how to install and use community Open MPI (version 3.1.1 is used as an example) on a BMS.

Perform the operations on each BMS in a cluster.

Prerequisites

Password-free login has been configured between BMSs in the cluster.

Procedure

Step 1 Install the HPC-X toolkit.

1. When community Open MPI is used, the Mellanox HPC-X toolkit is also required. The HPC-X version required by CentOS 7.3 is **hpcx-v2.2.0-gcc-MLNX_OFED_LINUX-4.3-1.0.1.0-redhat7.3-x86_64.tbz**.

   Download path: **https://developer.nvidia.com/networking/hpc-x**

2. Copy the downloaded software package to a directory, **/home/rhel** is recommended, on the BMS.
3. Run the following commands to decompress the HPC-X toolkit and change the toolkit directory:

   # tar -xvf hpcx-v2.2.0-gcc-MLNX_OFED_LINUX-4.3-1.0.1.0-redhat7.3-x86_64.tbz

   # mv hpcx-v2.2.0-gcc-MLNX_OFED_LINUX-4.3-1.0.1.0-redhat7.3-x86_64 /opt/hpcx-v2.2.0

Step 2 Install Open MPI.

1. Download community Open MPI of version **openmpi-3.1.0.tar.gz**.

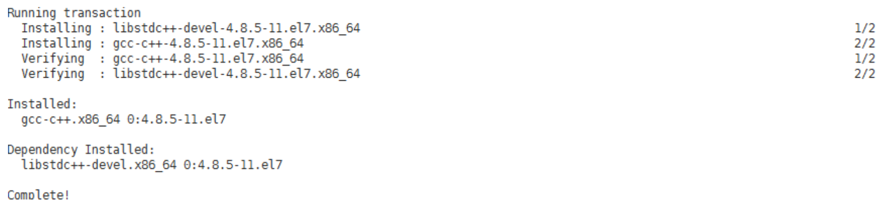
   Download path: **https://www.open-mpi.org/software/ompi/v3.1/**
2. Copy the downloaded software package to a directory, **/home/rhel** is recommended, on the BMS.
3. Run the following commands to decompress the software package:

   # tar -xzvf openmpi-3.1.0.tar.gz

   # cd openmpi-3.1.0
4. Install desired library files. Ensure that the BMS can access the Internet before the installation.
   a. Run the following command to install the dependency package:

      # yum install binutils-devel.x86_64 gcc-c++ autoconf automake libtool

      Figure 4-7 Successful installation of the dependency package

```
Running transaction
  Installing : libstdc++-devel-4.8.5-11.el7.x86_64                                                        1/2
  Installing : gcc-c++-4.8.5-11.el7.x86_64                                                                2/2
  Verifying  : gcc-c++-4.8.5-11.el7.x86_64                                                                1/2
  Verifying  : libstdc++-devel-4.8.5-11.el7.x86_64                                                        2/2

Installed:
  gcc-c++.x86_64 0:4.8.5-11.el7

Dependency Installed:
  libstdc++-devel.x86_64 0:4.8.5-11.el7

Complete!
```

5. Run the following commands to install and compile Open MPI:

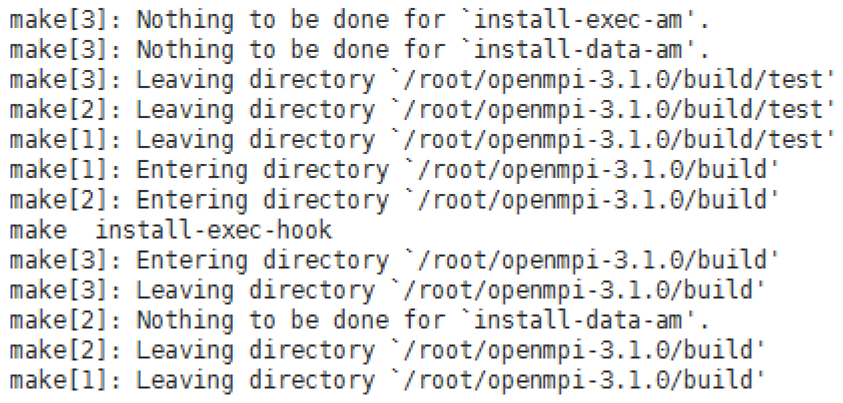
   # ./autogen.pl

   # mkdir build && cd build

   # ../configure --prefix=/opt/openmpi-310 --with-mxm=/opt/hpcx-v2.2.0/mxm

   # make all install

Figure 4-8 Successful installation of Open MPI



```
make[3]: Nothing to be done for `install-exec-am'.
make[3]: Nothing to be done for `install-data-am'.
make[3]: Leaving directory `/root/openmpi-3.1.0/build/test'
make[2]: Leaving directory `/root/openmpi-3.1.0/build/test'
make[1]: Leaving directory `/root/openmpi-3.1.0/build/test'
make[1]: Entering directory `/root/openmpi-3.1.0/build'
make[2]: Entering directory `/root/openmpi-3.1.0/build'
make  install-exec-hook
make[3]: Entering directory `/root/openmpi-3.1.0/build'
make[3]: Leaving directory `/root/openmpi-3.1.0/build'
make[2]: Nothing to be done for `install-data-am'.
make[2]: Leaving directory `/root/openmpi-3.1.0/build'
make[1]: Leaving directory `/root/openmpi-3.1.0/build'
```

Step 3 Configure MPI environment variables.

1. Add the following environment variables in **~/.bashrc** as a common user:

   export PATH=$PATH:/opt/openmpi-310/bin

   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/openmpi-310/lib

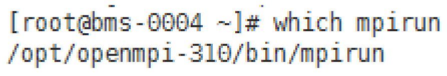
2. Run the following command to import MPI environment variables:

   $ source ~/.bashrc

3. Run the following command to check whether the MPI environment variables are correct:

   $ which mpirun

Figure 4-9 Correctly configured environment variables



```
[root@bms-0004 ~]# which mpirun
/opt/openmpi-310/bin/mpirun
```

If information shown in **Figure 4-9** is displayed, the environment variables have been correctly configured.

Step 4 Run community Open MPI on a BMS.

1. Run the following commands to generate an executable file:

   $ cd ~

   $ vi hello.c

   Edit the following content:

```
#include<mpi.h>
#include<stdio.h>
int main(int argc, char** argv){
        //Initialize the MPI environment
        MPI_Init(NULL, NULL);
        //Get the number of processes
        int world_size;
        MPI_Comm_size(MPI_COMM_WORLD, &world_size);
        //Get the rank of the process
        int world_rank;
        MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
        //Get the name of the processor
        char processor_name[MPI_MAX_PROCESSOR_NAME];
        int name_len;
        MPI_Get_processor_name(processor_name, &name_len);
        //Print off a hello world message.
        printf("Hello world from processor %s, rank %d"" out of %d processors\n",processor_name,
world_rank, world_size);
        //Finalize the MPI environment.
        MPI_Finalize();
    }
```

$ mpicc hello.c -o hello

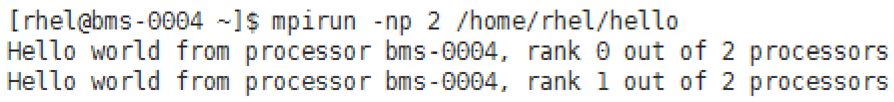
NOTICE

The **hello.c** file varies depending on MPI versions. You are required to re-compile the **hello.c** file by running the **mpicc hello.c -o hello** command, regardless of the MPI version.

2. Run the following command to run community Open MPI on a BMS:

   $ mpirun -np 2 /home/rhel/hello

   Figure 4-10 Successful execution of community Open MPI

```
[rhel@bms-0004 ~]$ mpirun -np 2 /home/rhel/hello
Hello world from processor bms-0004, rank 0 out of 2 processors
Hello world from processor bms-0004, rank 1 out of 2 processors
```

   If information shown in **Figure 4-10** is displayed, community Open MPI is running on the BMS.

----End

4.3.4 Spectrum MPI

Scenarios

This section describes how to install and use Spectrum MPI (version 10.01.01 is used as an example) in a BMS cluster.

Perform this operation on each BMS in a cluster.

Background

IBM Spectrum MPI v10.1 supports the following OSs:

- **IBM Spectrum MPI 10.1.0.1 Eval for x86_64 Linux**
  – Red Hat Enterprise Linux version 6.6 and later
  – Red Hat Enterprise Linux version 7.1 and later
  – SUSE Linux Enterprise Server version 11 SP4
  – SUSE Linux Enterprise Server version 12 and later
- **IBM Spectrum MPI 10.1.0.2 Eval for Power 8 Linux**

  Red Hat Enterprise Linux version 7.3 and later

Prerequisites

Password-free login has been configured between BMSs in the cluster.

Procedure

Step 1 Install IBM Spectrum MPI.

1. Obtain IBM Spectrum MPI software packages for registration.

   There are two IBM Spectrum MPI software packages, including the license and software:

   – ibm_smpi_lic_s-10.1Eval-rh7_Aug11.x86_64.rpm
   – ibm_smpi-10.01.01.0Eval-rh7_Aug11.x86_64.rpm

   Download path: **https://www-01.ibm.com/marketing/iwm/iwm/web/preLogin.do?source=swerpsysz-lsf-3**

2. Install IBM Spectrum MPI.

   a. Upload the obtained MPI software packages to a directory, **/home/rhel** is recommended, on the BMS running the MPI.

   b. Configure environment variables.

      ▪ If you choose to automatically accept the IBM Spectrum MPI installation license agreement, run the following command:

        # export IBM_SPECTRUM_MPI_LICENSE_ACCEPT=yes

      ▪ If you choose to manually accept the IBM Spectrum MPI installation license agreement, run the following command:

        # export IBM_SPECTRUM_MPI_LICENSE_ACCEPT=no

   c. Install the license.

      ▪ If you choose to automatically accept the IBM Spectrum MPI installation license agreement, run the following command:

        # rpm -ivh ibm_smpi_lic_s-10.1Eval-rh7_Aug11.x86_64.rpm

      ▪ If you choose to manually accept the IBM Spectrum MPI installation license agreement, run the following commands:

        # rpm -ivh ibm_smpi_lic_s-10.1Eval-rh7_Aug11.x86_64.rpm

        # sh /opt/ibm/spectrum_mpi/lap_se/bin/accept_spectrum_mpi_license.sh

   d. Install the software.

      # rpm -ivh ibm_smpi-10.01.01.0Eval-rh7_Aug11.x86_64.rpm

Step 2 Configure environment variables.

1. By default, Spectrum MPI is installed in the **/opt/ibm/spectrum_mpi** directory. In this case, you need to configure the following environment variables:

   $ export MPI_ROOT=/opt/ibm/spectrum_mpi

   $ export LD_LIBRARY_PATH=$MPI_ROOT/lib:$LD_LIBRARY_PATH

   $ export PATH=$MPI_ROOT/bin:$PATH

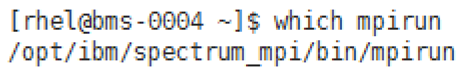
   $ export MANPATH=$MPI_ROOT/share/man:$MANPATH

   $ unset MPI_REMSH

2. Run the following command to check whether the MPI environment variables are correct:

$ which mpirun

Figure 4-11 Viewing Spectrum MPI environment variables



```
[rhel@bms-0004 ~]$ which mpirun
/opt/ibm/spectrum_mpi/bin/mpirun
```

Step 3 Run the executable file on a BMS through Spectrum MPI.

1. For example, the **hello.c** file is in the **/home/rhel/** directory, and the generated executable file is named **hello**. Then, run the following commands:

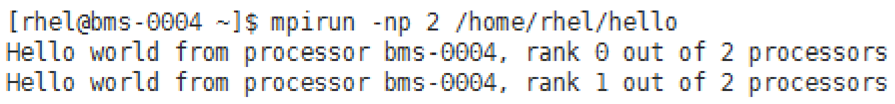
   $ cd /home/rhel/

   $ mpicc hello.c -o hello

2. Run the following command on a BMS to run the executable file through Spectrum MPI:

   $ mpirun -np 2 /home/rhel/hello

Figure 4-12 Successful Spectrum MPI execution on the BMS



```
[rhel@bms-0004 ~]$ mpirun -np 2 /home/rhel/hello
Hello world from processor bms-0004, rank 0 out of 2 processors
Hello world from processor bms-0004, rank 1 out of 2 processors
```

----End

4.3.5 Intel MPI

Scenarios

This section describes how to install and use Intel MPI (version l_mpi_2018.0.128 is used as an example) in a BMS cluster.

Perform this operation on each BMS in a cluster.

Prerequisites

Password-free login has been configured between BMSs in the cluster.

Procedure

Step 1 Install Intel MPI.

1. Download Intel MPI.

   Download path: **https://software.intel.com/en-us/intel-mpi-library**

2. Decompress and install Intel MPI.

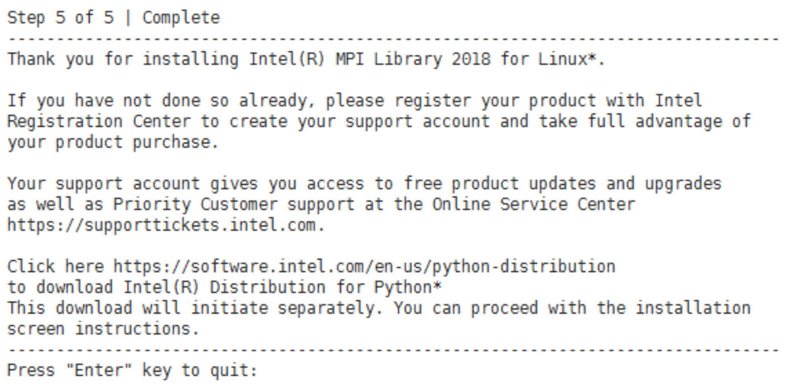
   For example, run the following commands to decompress and install **l_mpi_2018.0.128.tgz**:

   # tar -xvf l_mpi_2018.0.128.tgz

   # cd l_mpi_2018.0.128/

   # ./install.sh

Figure 4-13 Installed Intel MPI



```
Step 5 of 5 | Complete
--------------------------------------------------------------------------------
Thank you for installing Intel(R) MPI Library 2018 for Linux*.

If you have not done so already, please register your product with Intel
Registration Center to create your support account and take full advantage of
your product purchase.

Your support account gives you access to free product updates and upgrades
as well as Priority Customer support at the Online Service Center
https://supporttickets.intel.com.

Click here https://software.intel.com/en-us/python-distribution
to download Intel(R) Distribution for Python*
This download will initiate separately. You can proceed with the installation
screen instructions.
--------------------------------------------------------------------------------
Press "Enter" key to quit:
```

Step 2 Configure environment variables.

1. Add the following data to **~/.bashrc** as a common user:

   export PATH=$PATH:/opt/intel/impi/2018.0.128/bin64

   export LD_LIBRARY_PATH=/opt/intel/impi/2018.0.128/lib64

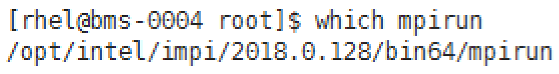
2. Run the following command to import environment variables:

   $ source ~/.bashrc

Step 3 Run the following command to check whether environment variables have been imported:

$ which mpirun

Figure 4-14 Successful importing of Intel MPI environment variables



```
[rhel@bms-0004 root]$ which mpirun
/opt/intel/impi/2018.0.128/bin64/mpirun
```

If information shown in **Figure 4-14** is displayed, the environment variables have been imported.

Step 4 Run Intel MPI on a BMS.

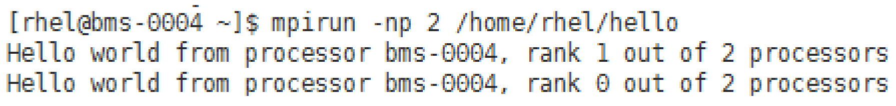
1. Run the following command to generate an executable file:

   $ mpicc hello.c -o hello

2. Run the following command to run Intel MPI on a BMS:

   $ mpirun -np 2 /home/rhel/hello

Figure 4-15 Running Intel MPI on a BMS



```
[rhel@bms-0004 ~]$ mpirun -np 2 /home/rhel/hello
Hello world from processor bms-0004, rank 1 out of 2 processors
Hello world from processor bms-0004, rank 0 out of 2 processors
```

----End

4.3.6 Platform MPI

Scenarios

This section describes how to install and use Platform MPI (version platform_mpi-09.01.04.03r-ce.bin is used as an example) in a BMS cluster.

Perform this operation on each BMS in a cluster.

Prerequisites

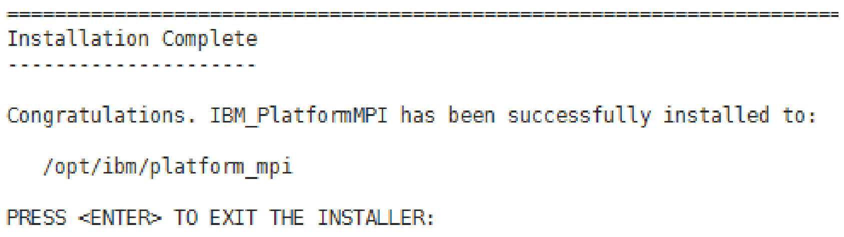
Password-free login has been configured between BMSs in the cluster.

Procedure

Step 1 Install Platform MPI.

1. Download the software package, such as **platform_mpi- 09.01.04.03r-ce.bin**.
2. Run the following command to install the dependency package:
   # yum install glibc.i686 libgcc-4.8.5-11.el7.i686 libgcc_s.so.1
3. Run the following command to assign execution permissions:
   #chmod +x platform_mpi-09.01.04.03r-ce.bin
4. Install Platform MPI.
   # ./platform_mpi-09.01.04.03r-ce.bin
   Platform MPI is installed in **/opt/ibm/platform_mpi** by default.

Figure 4-16 Installed Platform MPI

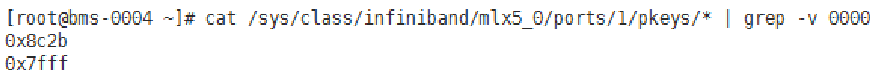


```
===============================================================================
Installation Complete
---------------------

Congratulations. IBM_PlatformMPI has been successfully installed to:

   /opt/ibm/platform_mpi

PRESS <ENTER> TO EXIT THE INSTALLER:
```

Step 2 Configure environment variables.

1. Run the following command to obtain the pkey:
   # cat /sys/class/infiniband/mlx5_0/ports/1/pkeys/* | grep -v 0000

Figure 4-17 Obtaining the pkey



```
[root@bms-0004 ~]# cat /sys/class/infiniband/mlx5_0/ports/1/pkeys/* | grep -v 0000
0x8c2b
0x7fff
```

2. Add the following data to **~/.bashrc** as a common user:
   export MPI_ROOT=/opt/ibm/platform_mpi
   export PATH=$MPI_ROOT/bin:$PATH
   export LD_LIBRARY_PATH=/opt/ibm/platform_mpi/lib/linux_amd64
   export MPI_IB_PKEY=*pkey obtained in* ***Step 2.1***
   $source ~/.bashrc

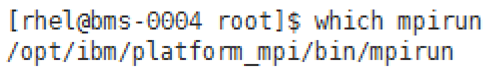
NOTE

If there are multiple pkeys, use a comma to separate them.

3. Run the following command to check whether the environment variables have been configured:

   # which mpirun

   Figure 4-18 Viewing the environment variables

```
[rhel@bms-0004 root]$ which mpirun
/opt/ibm/platform_mpi/bin/mpirun
```

Step 3 Run Platform MPI on a BMS.

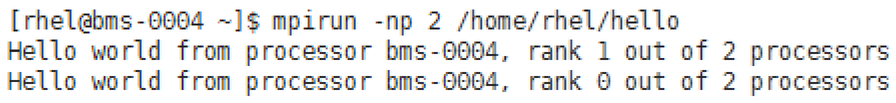
1. Run the following command to re-compile the **hello.c** file:

   $ mpicc hello.c -o hello

2. Run the following command to run Platform MPI on a BMS:

   $ mpirun -np 2 /home/rhel/hello

   Figure 4-19 Running Platform MPI on a BMS

```
[rhel@bms-0004 ~]$ mpirun -np 2 /home/rhel/hello
Hello world from processor bms-0004, rank 1 out of 2 processors
Hello world from processor bms-0004, rank 0 out of 2 processors
```

----End

4.4 Installing and Using MPIs (Kunpeng BMS)

This section uses the CentOS 7.6 OS as an example to describe how to run MPIs on a single Kunpeng BMS.

4.4.1 MPIs Supported in the Kunpeng BMS Scenario

The following MPIs are supported in the Kunpeng BMS scenario:

- Built-in OpenMPI of the IB driver
- Community Open MPI
- MPICH

Install and use an MPI as needed.

4.4.2 Open MPI Delivered with the IB Driver

Scenarios

This section describes how to install and use Open MPI (version 4.0.2a1 is used as an example) delivered with the IB driver on a BMS.

Perform the operations on each BMS in a cluster.

Prerequisites

Password-free login has been configured between BMSs in the cluster.

Procedure

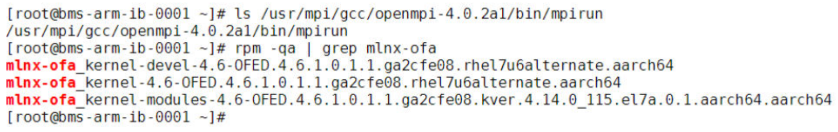
Step 1 Check whether the IB driver has been installed.

1. Run the following commands to check whether the IB driver has been installed:

   $ ls /usr/mpi/gcc/openmpi-4.0.2a1/bin/mpirun

   $ rpm -qa | grep mlnx-ofa

   Figure 4-20 Checking the IB driver

```
[root@bms-arm-ib-0001 ~]# ls /usr/mpi/gcc/openmpi-4.0.2a1/bin/mpirun
/usr/mpi/gcc/openmpi-4.0.2a1/bin/mpirun
[root@bms-arm-ib-0001 ~]# rpm -qa | grep mlnx-ofa
mlnx-ofa_kernel-devel-4.6-OFED.4.6.1.0.1.1.ga2cfe08.rhel7u6alternate.aarch64
mlnx-ofa_kernel-4.6-OFED.4.6.1.0.1.1.ga2cfe08.rhel7u6alternate.aarch64
mlnx-ofa_kernel-modules-4.6-OFED.4.6.1.0.1.1.ga2cfe08.kver.4.14.0_115.el7a.0.1.aarch64.aarch64
[root@bms-arm-ib-0001 ~]#
```

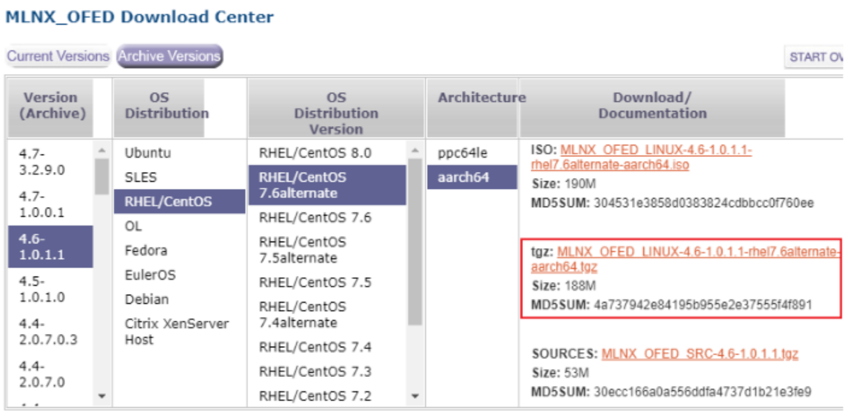
2. Check the command output.
   - If information shown in **Figure 4-20** is displayed, the IB driver has been installed. Then, go to **Step 3**.
   - If the IB driver has not been installed, go to **Step 2**.

Step 2 Install the IB driver.

1. Download the installation package **MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6alternate-aarch64.tgz**.

   Download path: **https://network.nvidia.com/products/infiniband-drivers/linux/mlnx_ofed/**

   Figure 4-21 IB driver download center

Version (Archive)	OS Distribution	OS Distribution Version	Architecture	Download/ Documentation
4.7-3.2.9.0	Ubuntu	RHEL/CentOS 8.0	ppc64le	ISO: MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6alternate-aarch64.iso Size: 190M MD5SUM: 304531e3858d0383824cdbbcc0f760ee
4.7-1.0.0.1	SLES	RHEL/CentOS 7.6alternate	aarch64	tgz: MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6alternate-aarch64.tgz Size: 188M MD5SUM: 4a737942e84195b955e2e37555f4f891
4.6-1.0.1.1	RHEL/CentOS	RHEL/CentOS 7.6		SOURCES: MLNX_OFED_SRC-4.6-1.0.1.1.tgz Size: 53M MD5SUM: 30ecc166a0a556ddfa4737d1b21e3fe9
4.5-1.0.1.0	OL	RHEL/CentOS 7.5alternate		
4.4-2.0.7.0.3	Fedora	RHEL/CentOS 7.5		
4.4-2.0.7.0	EulerOS	RHEL/CentOS 7.4alternate		
	Debian	RHEL/CentOS 7.4		
	Citrix XenServer Host	RHEL/CentOS 7.3		
		RHEL/CentOS 7.2		

2. Run the following commands to install the software package:

   # yum install tk tcl -y

   # tar -xvf MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6alternate-aarch64.tgz

   # cd MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6-x86_64/

   # ./mlnxofedinstall

Step 3 Install and configure UCX.

1. Download the UCX installation package.

   # cd /opt && wget https://github.com/openucx/ucx/releases/download/v1.6.0/ucx-1.6.0.tar.gz

2. Decompress the package.

   # tar -xvf ucx-1.6.0.tar.gz

3. Compile and install UCX.

# cd /opt/ucx-1.6.0

# yum install autoconf automake libtool numactl-devel -y

# ./contrib/configure-release --prefix=/opt/ucx160 --enable-optimizations

# make && make install

Step 4 Configure UCX.

1. Create non-root user **rhel**.

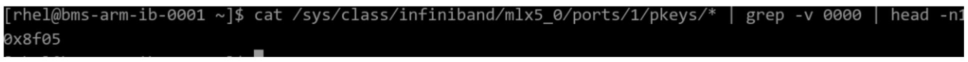
   # useradd rhel; su - rhel

2. Obtain PKEY, delete the third digit of PKEY, and use this value to replace {pkey} in **Step 4.4**.

   # cat /sys/class/infiniband/mlx5_0/ports/1/pkeys/* | grep -v 0000 | head -n1

   For example, if the obtained PKEY is 0x8f05, delete the third digit to obtain 0xf05.

   Figure 4-22 Obtaining PKEY



```
[rhel@bms-arm-ib-0001 ~]$ cat /sys/class/infiniband/mlx5_0/ports/1/pkeys/* | grep -v 0000 | head -n1
0x8f05
```

3. Obtain UCX PKEY.

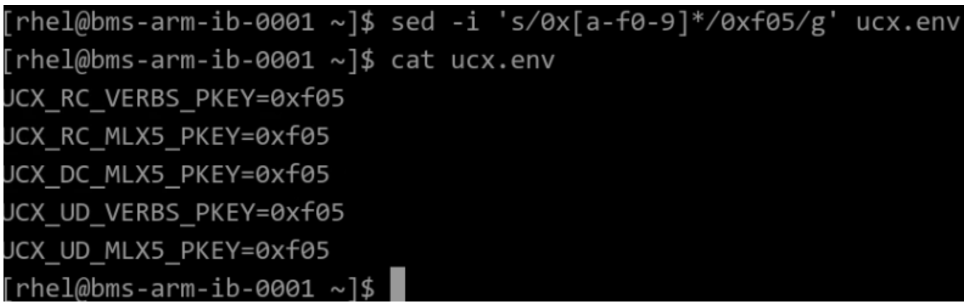
   # ucx_info -c | grep -i pkey > ucx.env

4. Replace PKEY in UCX.

   # sed -i 's/0x[a-f0-9]*/{pkey}/g' ucx.env

   In this example, run **sed -i 's/0x[a-f0-9]*/0xf05/g' ucx.env**.

   Figure 4-23 Replacing PKEY in UCX



```
[rhel@bms-arm-ib-0001 ~]$ sed -i 's/0x[a-f0-9]*/0xf05/g' ucx.env
[rhel@bms-arm-ib-0001 ~]$ cat ucx.env
UCX_RC_VERBS_PKEY=0xf05
UCX_RC_MLX5_PKEY=0xf05
UCX_DC_MLX5_PKEY=0xf05
UCX_UD_VERBS_PKEY=0xf05
UCX_UD_MLX5_PKEY=0xf05
[rhel@bms-arm-ib-0001 ~]$
```

5. Set UCX PKEY as an environment variable.

   # sed -i 's/^UCX/export UCX/g' ucx.env

   # cat ucx.env >> ~/.bashrc

Step 5 Configure MPI environment variables.

1. Use VIM to open the **~/.bashrc** file and add the following data to the file:

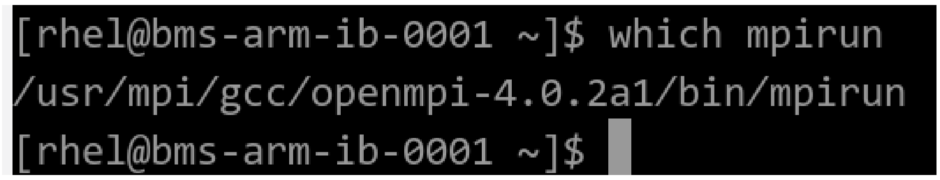
   export PATH=$PATH:/usr/mpi/gcc/openmpi-4.0.2a1/bin

   export LD_LIBRARY_PATH=/usr/mpi/gcc/openmpi-4.0.2a1/lib64

2. Run the following command to check whether the MPI environment variables are correct:

   $ which mpirun

   Figure 4-24 Viewing the Open MPI environment variables

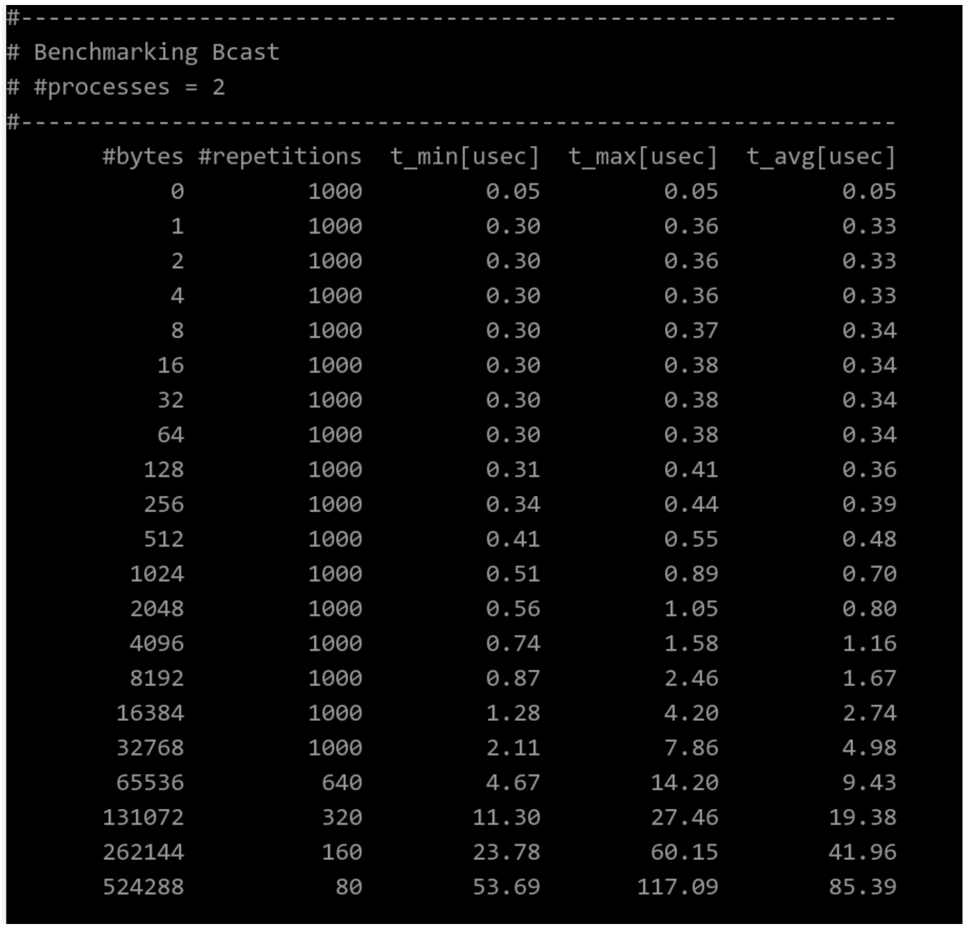
```
[rhel@bms-arm-ib-0001 ~]$ which mpirun
/usr/mpi/gcc/openmpi-4.0.2a1/bin/mpirun
[rhel@bms-arm-ib-0001 ~]$
```

   If information shown in **Figure 4-24** is displayed, environment variables have been configured.

3. Run the following command to run Open MPI delivered with the IB driver on a BMS:

   #mpirun -np 2 -mca btl_openib_if_include "mlx5_0:1" -x MXM_IB_USE_GRH=y /usr/mpi/gcc/openmpi-3.1.0rc2/tests/imb/IMB-MPI1 PingPong

   Figure 4-25 Running Open MPI delivered with the IB driver

```
#-----------------------------------------------------------------
# Benchmarking Bcast
# #processes = 2
#-----------------------------------------------------------------
       #bytes #repetitions  t_min[usec]  t_max[usec]  t_avg[usec]
            0         1000         0.05         0.05         0.05
            1         1000         0.30         0.36         0.33
            2         1000         0.30         0.36         0.33
            4         1000         0.30         0.36         0.33
            8         1000         0.30         0.37         0.34
           16         1000         0.30         0.38         0.34
           32         1000         0.30         0.38         0.34
           64         1000         0.30         0.38         0.34
          128         1000         0.31         0.41         0.36
          256         1000         0.34         0.44         0.39
          512         1000         0.41         0.55         0.48
         1024         1000         0.51         0.89         0.70
         2048         1000         0.56         1.05         0.80
         4096         1000         0.74         1.58         1.16
         8192         1000         0.87         2.46         1.67
        16384         1000         1.28         4.20         2.74
        32768         1000         2.11         7.86         4.98
        65536          640         4.67        14.20         9.43
       131072          320        11.30        27.46        19.38
       262144          160        23.78        60.15        41.96
       524288           80        53.69       117.09        85.39
```

----End

4.4.3 Community Open MPI

Scenarios

This section describes how to install and use community Open MPI (version 4.0.2 is used as an example) on a BMS.

Perform the operations on each BMS in a cluster.

Prerequisites

Password-free login has been configured between BMSs in the cluster.

Procedure

Step 1 Install Open MPI.

1. Download community Open MPI of version **openmpi-4.0.2.tar.bz2**.

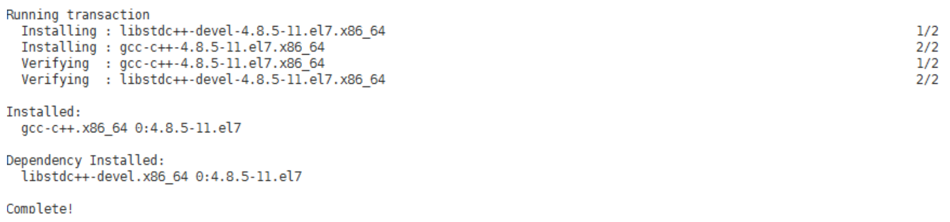
   Download path: **https://download.open-mpi.org/release/open-mpi/v4.0/openmpi-4.0.2.tar.bz2**
2. Copy the downloaded software package to a directory, **/home/rhel** is recommended, on the BMS.
3. Run the following commands to decompress the software package:

   # tar -xzvf openmpi-4.0.2.tar.bz2

   # cd openmpi-4.0.2
4. Install desired dependency packages. Ensure that the BMS can access the Internet before the installation.

   # yum install binutils-devel.x86_64 gcc-c++ autoconf automake libtool

   Figure 4-26 Successful installation of dependency packages

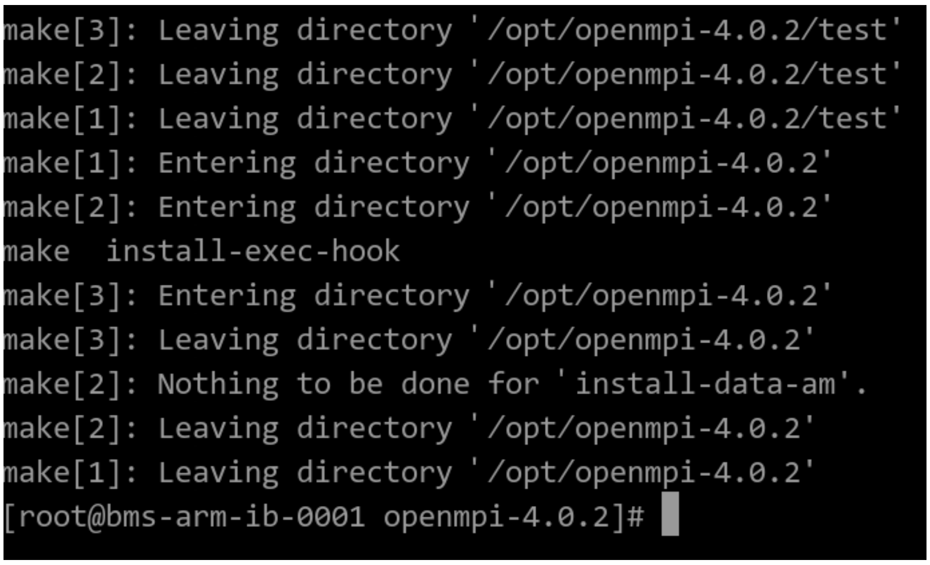
```
Running transaction
  Installing : libstdc++-devel-4.8.5-11.el7.x86_64                                                     1/2
  Installing : gcc-c++-4.8.5-11.el7.x86_64                                                             2/2
  Verifying  : gcc-c++-4.8.5-11.el7.x86_64                                                             1/2
  Verifying  : libstdc++-devel-4.8.5-11.el7.x86_64                                                     2/2

Installed:
  gcc-c++.x86_64 0:4.8.5-11.el7

Dependency Installed:
  libstdc++-devel.x86_64 0:4.8.5-11.el7

Complete!
```

5. Run the following commands to install and compile Open MPI:

   # ./openmpi-4.0.2/configure --prefix=/opt/openmpi-402--enable-mpirun-prefix-by-default --enable-mpi1-compatibility --with-ucx=/opt/ucx160

   # make -j 128 && make install

Figure 4-27 Successful installation of Open MPI



```
make[3]: Leaving directory '/opt/openmpi-4.0.2/test'
make[2]: Leaving directory '/opt/openmpi-4.0.2/test'
make[1]: Leaving directory '/opt/openmpi-4.0.2/test'
make[1]: Entering directory '/opt/openmpi-4.0.2'
make[2]: Entering directory '/opt/openmpi-4.0.2'
make  install-exec-hook
make[3]: Entering directory '/opt/openmpi-4.0.2'
make[3]: Leaving directory '/opt/openmpi-4.0.2'
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/opt/openmpi-4.0.2'
make[1]: Leaving directory '/opt/openmpi-4.0.2'
[root@bms-arm-ib-0001 openmpi-4.0.2]#
```

Step 2 Configure MPI environment variables.

1. Add the following environment variables in **~/.bashrc** as a common user:

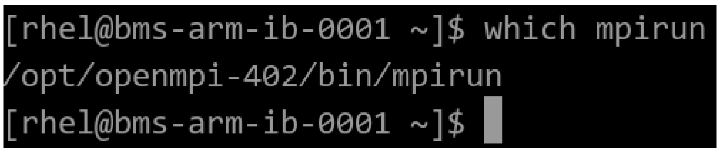
   export PATH=$PATH:/opt/openmpi-310/bin

   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/openmpi-310/lib
2. Run the following command to import MPI environment variables:

   $ source ~/.bashrc
3. Run the following command to check whether the MPI environment variables are correct:

   $ which mpirun

   Figure 4-28 Correctly configured environment variables

```
[rhel@bms-arm-ib-0001 ~]$ which mpirun
/opt/openmpi-402/bin/mpirun
[rhel@bms-arm-ib-0001 ~]$
```

   If information shown in **Figure 4-28** is displayed, the environment variables have been correctly configured.

Step 3 Run community Open MPI on a BMS.

1. Run the following commands to generate an executable file:

   $ cd ~

   $ vi hello.c

   Edit the following content:

```
#include<mpi.h>
#include<stdio.h>
int main(int argc, char** argv){
       //Initialize the MPI environment
       MPI_Init(NULL, NULL);
       //Get the number of processes
```

```
      int world_size;
      MPI_Comm_size(MPI_COMM_WORLD, &world_size);
      //Get the rank of the process
      int world_rank;
      MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
      //Get the name of the processor
      char processor_name[MPI_MAX_PROCESSOR_NAME];
      int name_len;
      MPI_Get_processor_name(processor_name, &name_len);
      //Print off a hello world message.
      printf("Hello world from processor %s, rank %d"" out of %d processors\n",processor_name,
world_rank, world_size);
      //Finalize the MPI environment.
      MPI_Finalize();
   }
```

$ mpicc hello.c -o hello

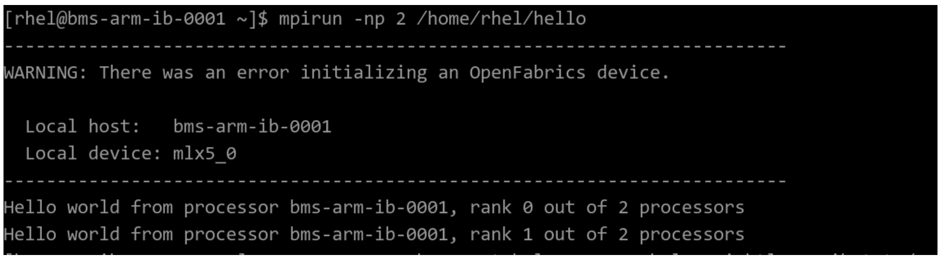
NOTICE

The **hello.c** file varies depending on MPI versions. You are required to re-compile the **hello.c** file by running the **mpicc hello.c -o hello** command, regardless of the MPI version.

2. Run community Open MPI on a BMS.

   $ mpirun -np 2 /home/rhel/hello

   Figure 4-29 Successful execution of community Open MPI



```
[rhel@bms-arm-ib-0001 ~]$ mpirun -np 2 /home/rhel/hello
--------------------------------------------------------------------------
WARNING: There was an error initializing an OpenFabrics device.

  Local host:   bms-arm-ib-0001
  Local device: mlx5_0
--------------------------------------------------------------------------
Hello world from processor bms-arm-ib-0001, rank 0 out of 2 processors
Hello world from processor bms-arm-ib-0001, rank 1 out of 2 processors
```

   If information shown in **Figure 4-29** is displayed, community Open MPI is running on the BMS.

----End

4.4.4 MPICH

Scenarios

This section describes how to install and use MPICH (version mpich-3.3.2 is used as an example) in a Kunpeng BMS cluster.

Perform the operations on each BMS in a cluster.

Prerequisites

Password-free login has been configured between BMSs in the cluster.

Procedure

Step 1 Install MPICH.

1. Download MPICH.

   Download path: https://aur.archlinux.org/packages/mpich/

2. Decompress the installation package and install MPICH.

   For example, if the installation package is **mpich-3.3.2.tar.gz**, run the following commands:

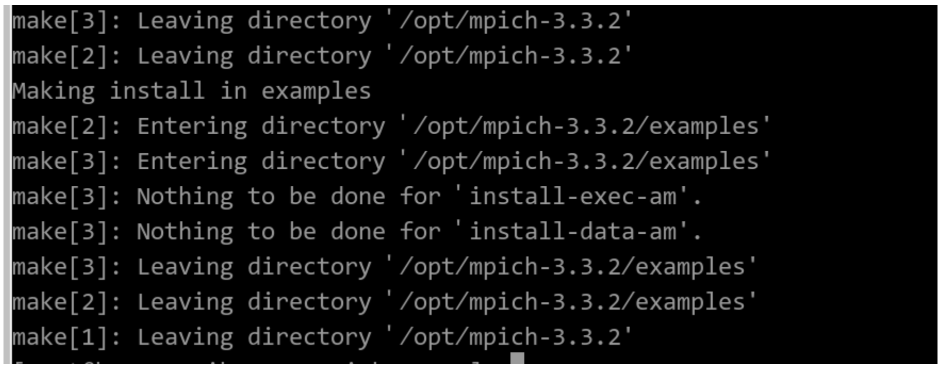
   # tar -xvf mpich-3.3.2.tar.gz

   # cd mpich-3.3.2/

   # ./configure --prefix=/opt/mpich-332 --with-device=ch4:ucx --with-ucx=/pub/mpi/ucx160/ --enable-fast=O3 CFLAGS="-fPIC -std=gnu11" FFLAGS=-fPIC CXXFLAGS=-fPIC FCFLAGS=-fPIC

   # make -j 128 && make install

   Figure 4-30 Successfully installed MPCHI



```
make[3]: Leaving directory '/opt/mpich-3.3.2'
make[2]: Leaving directory '/opt/mpich-3.3.2'
Making install in examples
make[2]: Entering directory '/opt/mpich-3.3.2/examples'
make[3]: Entering directory '/opt/mpich-3.3.2/examples'
make[3]: Nothing to be done for 'install-exec-am'.
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/opt/mpich-3.3.2/examples'
make[2]: Leaving directory '/opt/mpich-3.3.2/examples'
make[1]: Leaving directory '/opt/mpich-3.3.2'
```

Step 2 Configure environment variables.

1. Add the following content to **~/.bashrc** as a common user:

   export PATH=/opt/mpich-332/bin: $PATH

   export LD_LIBRARY_PATH=/opt/mpich-332/lib

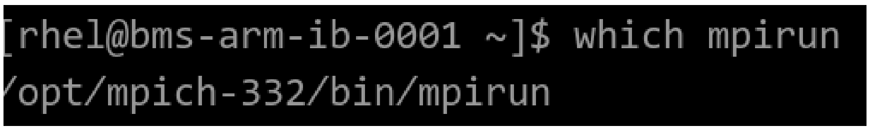
2. Run the following command to import environment variables:

   $ source ~/.bashrc

Step 3 Run the following command to check whether environment variables have been imported:

$ which mpirun

Figure 4-31 Successful imported MPICH environment variables



```
[rhel@bms-arm-ib-0001 ~]$ which mpirun
/opt/mpich-332/bin/mpirun
```

If information shown in **Figure 4-31** is displayed, the environment variables have been imported.

Step 4 Run MPICH on a BMS.

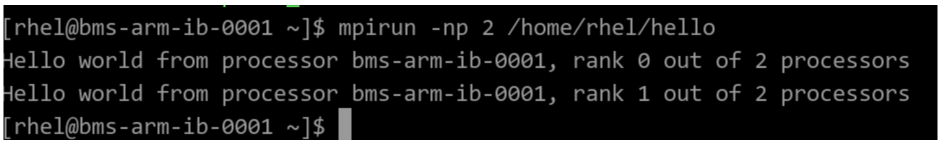
1. Run the following command to generate an executable file:

   $ mpicc hello.c -o hello

2. Run the following command to run MPICH:

   $ mpirun -np 2 /home/rhel/hello

Figure 4-32 Running MPICH on a BMS



```
[rhel@bms-arm-ib-0001 ~]$ mpirun -np 2 /home/rhel/hello
Hello world from processor bms-arm-ib-0001, rank 0 out of 2 processors
Hello world from processor bms-arm-ib-0001, rank 1 out of 2 processors
[rhel@bms-arm-ib-0001 ~]$
```

----End

4.5 Running MPI Applications in an HPC Cluster (x86 BMS)

This section uses the CentOS 7.3 OS as an example to describe how to run MPIs in a cluster.

4.5.1 Open MPI Delivered with the IB Driver

Scenarios

This section describes how to run Open MPI (version 3.1.0rc2 is used as an example) delivered with the IB driver in a BMS cluster.

Prerequisites

- Password-free login has been configured between BMSs in the cluster.
- Open MPI delivered with the IB driver has been installed on all BMSs in the cluster.

Procedure

Step 1 Disable the firewall.

1. Log in to a BMS in the cluster.
2. Run the following command to disable the BMS firewall:

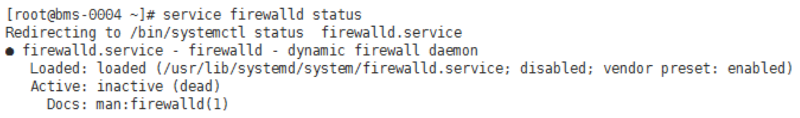
   # service firewalld stop

   # iptables -F

3. Run the following command to check whether the firewall has been disabled:

   # service firewalld status

Figure 4-33 Disabled firewall



```
[root@bms-0004 ~]# service firewalld status
Redirecting to /bin/systemctl status  firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

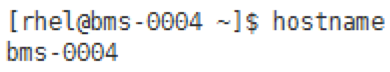
4. Log in to all other BMSs in the cluster and repeat **Step 1.2** to **Step 1.3** to disable firewalls on all BMSs.

Step 2 Modify the configuration file.

1. Log in to a BMS in the cluster.
2. Run the following command to obtain the BMS hostname:

   $ hostname

   Figure 4-34 Viewing the BMS hostname

   ```
   [rhel@bms-0004 ~]$ hostname
   bms-0004
   ```
3. Log in to all other BMSs in the cluster and repeat **Step 2.1** to **Step 2.2** to obtain hostnames of all BMSs.
4. Log in to a BMS in the cluster.
5. Run the following command to add the hosts configuration file:

   # vi /etc/hosts

   Add the private network IP addresses and hostnames of all BMSs in the cluster. For example, run the following commands:

   192.168.0.1 bms-0004

   192.168.0.2 bms-0005

   ...
6. Run the following command to add the **hostfile** file:

   $vi hostfile

   Add hostnames of all BMSs in the cluster, for example:

   bms-0004

   bms-0005

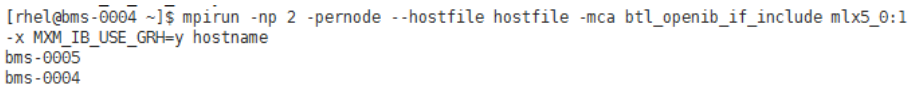
   ...
7. Log in to all BMSs in the cluster and repeat **Step 2.5** to **Step 2.6**.

Step 3 Run MPI benchmark.

1. Perform the following command on any BMS to check whether the **hostfile** file has been configured:

   $ mpirun -np 2 -pernode --hostfile hostfile -mca btl_openib_if_include "mlx5_0:1" -x MXM_IB_USE_GRH=y hostname

   Figure 4-35 Checking the configuration file

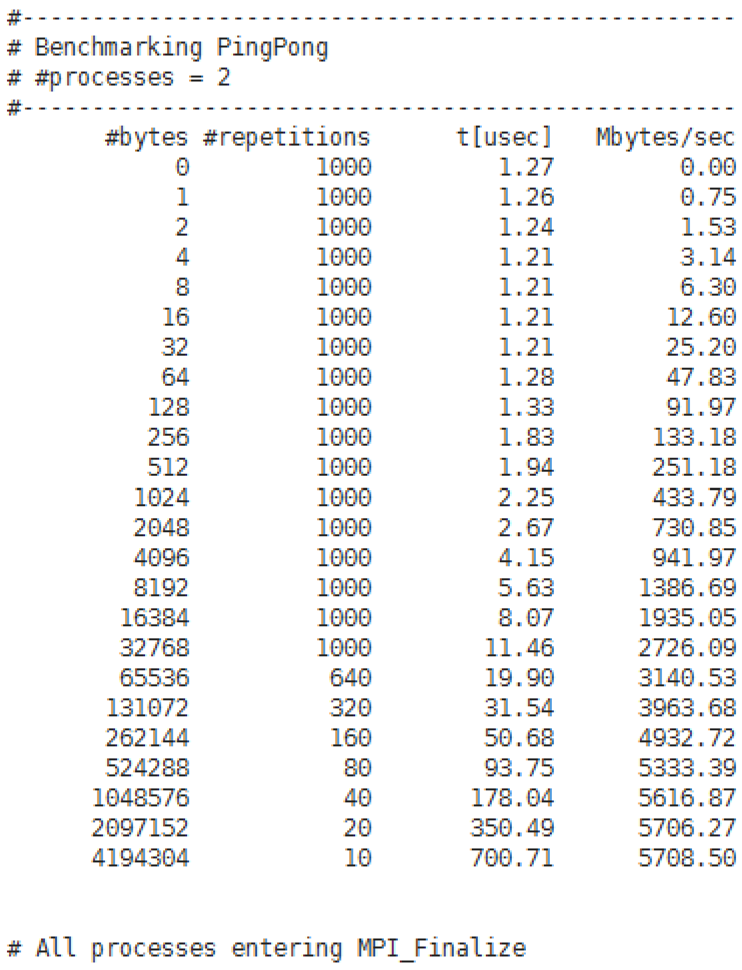
   ```
   [rhel@bms-0004 ~]$ mpirun -np 2 -pernode --hostfile hostfile -mca btl_openib_if_include mlx5_0:1
   -x MXM_IB_USE_GRH=y hostname
   bms-0005
   bms-0004
   ```

   If the hostnames of all BMSs in the cluster are displayed, as shown in **Figure 4-35**, the **hostfile** file has been configured.
2. Run the MPI benchmark on any BMS with the hostfile path specified.

   For example, there are two BMSs in the cluster. Then, run the following command:

$ mpirun -np 2 -pernode --hostfile hostfile -mca btl_openib_if_include "mlx5_0:1" -x MXM_IB_USE_GRH=y /usr/mpi/gcc/openmpi-3.1.0rc2/tests/imb/IMB-MPI1 PingPong

Figure 4-36 Running Open MPI delivered with the IB drivers in the cluster



```
#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec
            0         1000         1.27         0.00
            1         1000         1.26         0.75
            2         1000         1.24         1.53
            4         1000         1.21         3.14
            8         1000         1.21         6.30
           16         1000         1.21        12.60
           32         1000         1.21        25.20
           64         1000         1.28        47.83
          128         1000         1.33        91.97
          256         1000         1.83       133.18
          512         1000         1.94       251.18
         1024         1000         2.25       433.79
         2048         1000         2.67       730.85
         4096         1000         4.15       941.97
         8192         1000         5.63      1386.69
        16384         1000         8.07      1935.05
        32768         1000        11.46      2726.09
        65536          640        19.90      3140.53
       131072          320        31.54      3963.68
       262144          160        50.68      4932.72
       524288           80        93.75      5333.39
      1048576           40       178.04      5616.87
      2097152           20       350.49      5706.27
      4194304           10       700.71      5708.50


# All processes entering MPI_Finalize
```

If information shown in **Figure 4-36** is displayed, Open MPI delivered with the IB driver is running in the cluster.

----End

4.5.2 Community Open MPI

Scenarios

This section describes how to run community Open MPI (version 3.1.1 is used as an example) in a BMS cluster.

Prerequisites

- Password-free login has been configured between BMSs in the cluster.
- Community Open MPI has been installed on all BMSs in the cluster.

Procedure

Step 1 Disable the firewall.

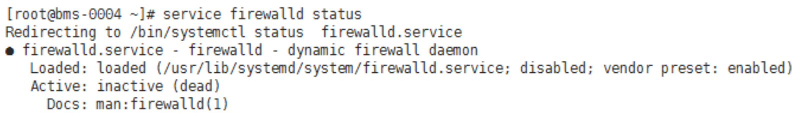
1. Log in to a BMS in the cluster.
2. Run the following command to disable the BMS firewall:

   # service firewalld stop

   # iptables -F
3. Run the following command to check whether the firewall has been disabled:

   # service firewalld status

   Figure 4-37 Disabled firewall

```
[root@bms-0004 ~]# service firewalld status
Redirecting to /bin/systemctl status  firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

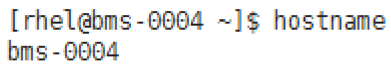
4. Log in to all other BMSs in the cluster and repeat **Step 1.2** to **Step 1.3** to disable firewalls on all BMSs.

Step 2 Modify the configuration file.

1. Log in to a BMS in the cluster.
2. Run the following command to obtain the BMS hostname:

   $ hostname

   Figure 4-38 Viewing the BMS hostname

```
[rhel@bms-0004 ~]$ hostname
bms-0004
```

3. Log in to all other BMSs in the cluster and repeat **Step 2.1** to **Step 2.2** to obtain hostnames of all BMSs.
4. Log in to a BMS in the cluster.
5. Run the following command to add the hosts configuration file:

   # vi /etc/hosts

   Add the private network IP addresses and hostnames of all BMSs in the cluster. For example, run the following commands:

   192.168.0.1 bms-0004

   192.168.0.2 bms-0005

   ...
6. Run the following command to add the **hostfile** file:

   $vi hostfile

   Add hostnames of all BMSs in the cluster, for example:

   bms-0004

   bms-0005

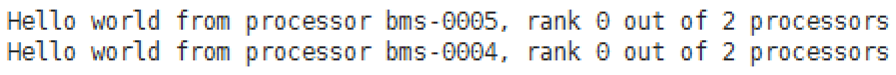
   ...
7. Log in to all BMSs in the cluster and repeat **Step 2.5** to **Step 2.6**.

Step 3 Log in to any BMS and run the community Open MPI.

For example, there are two BMSs in the cluster. Then, run the following command:

$ mpirun -np 2 --pernode -hostfile hostfile /home/rhel/hello

Figure 4-39 Successful execution of community Open MPI in the cluster



```
Hello world from processor bms-0005, rank 0 out of 2 processors
Hello world from processor bms-0004, rank 0 out of 2 processors
```

NOTE

Specify the path of **hostfile** when running it. The path of the executable file **hello** must be absolute. All executable files in the cluster must be in the same directory.

----End

4.5.3 Spectrum MPI

Scenarios

This section describes how to run Spectrum MPI (version 10.01.01 is used as an example) in a BMS cluster.

Prerequisites

- Password-free login has been configured between BMSs in the cluster.
- Spectrum MPI has been installed on all BMSs in the cluster.

Procedure

Step 1 Disable the firewall.

1. Log in to a BMS in the cluster.
2. Run the following command to disable the BMS firewall:

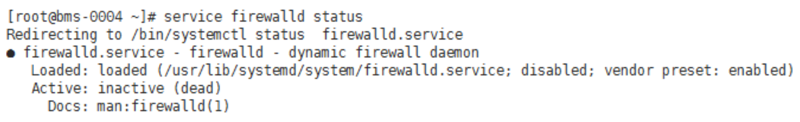
   # service firewalld stop

   # iptables -F

3. Run the following command to check whether the firewall has been disabled:

   # service firewalld status

   Figure 4-40 Disabled firewall

```
[root@bms-0004 ~]# service firewalld status
Redirecting to /bin/systemctl status  firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

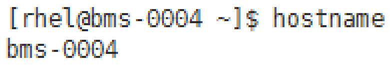
4. Log in to all other BMSs in the cluster and repeat **Step 1.2** to **Step 1.3** to disable firewalls on all BMSs.

Step 2 Modify the configuration file.

1. Log in to a BMS in the cluster.
2. Run the following command to obtain the BMS hostname:

   $ hostname

Figure 4-41 Viewing the BMS hostname



```
[rhel@bms-0004 ~]$ hostname
bms-0004
```

3. Log in to all other BMSs in the cluster and repeat **Step 2.1** to **Step 2.2** to obtain hostnames of all BMSs.
4. Log in to a BMS in the cluster.
5. Run the following command to add the hosts configuration file:

   # vi /etc/hosts

   Add the private network IP addresses and hostnames of all BMSs in the cluster. For example, run the following commands:

   192.168.0.1 bms-0004

   192.168.0.2 bms-0005

   ...
6. Run the following command to add the **hostfile** file:

   $vi hostfile

   Add hostnames of all BMSs in the cluster, for example:

   bms-0004

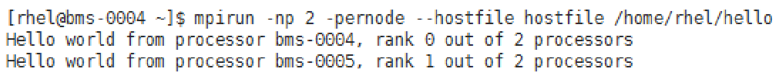
   bms-0005

   ...
7. Log in to all BMSs in the cluster and repeat **Step 2.5** to **Step 2.6**.

Step 3 Run the following command on a BMS to run the executable file through Spectrum MPI:

$ mpirun -np 2 -pernode --hostfile hostfile /home/rhel/hello

Figure 4-42 Successful execution of Spectrum MPI in the BMS cluster



```
[rhel@bms-0004 ~]$ mpirun -np 2 -pernode --hostfile hostfile /home/rhel/hello
Hello world from processor bms-0004, rank 0 out of 2 processors
Hello world from processor bms-0005, rank 1 out of 2 processors
```

NOTE

Specify the path of **hostfile** when running it. The path of the executable file **hello** must be absolute. All executable files in the cluster must be in the same directory.

----End

4.5.4 Intel MPI

Scenarios

This section describes how to run Intel MPI (version l_mpi_2017.3.196 is used as an example) in a BMS cluster.

Prerequisites

- Password-free login has been configured between BMSs in the cluster.

- Spectrum MPI has been installed on all BMSs in the cluster.

Procedure

Step 1 Disable the firewall.

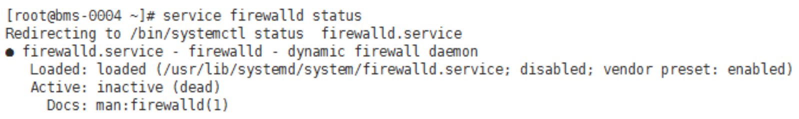
1. Log in to a BMS in the cluster.
2. Run the following command to disable the BMS firewall:

   # service firewalld stop

   # iptables -F
3. Run the following command to check whether the firewall has been disabled:

   # service firewalld status

   Figure 4-43 Disabled firewall

```
[root@bms-0004 ~]# service firewalld status
Redirecting to /bin/systemctl status  firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

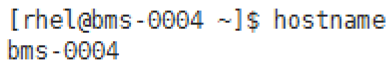
4. Log in to all other BMSs in the cluster and repeat **Step 1.2** to **Step 1.3** to disable firewalls on all BMSs.

Step 2 Modify the configuration file.

1. Log in to a BMS in the cluster.
2. Run the following command to obtain the BMS hostname:

   $ hostname

   Figure 4-44 Viewing the BMS hostname

```
[rhel@bms-0004 ~]$ hostname
bms-0004
```

3. Log in to all other BMSs in the cluster and repeat **Step 2.1** to **Step 2.2** to obtain hostnames of all BMSs.
4. Log in to a BMS in the cluster.
5. Run the following command to add the hosts configuration file:

   # vi /etc/hosts

   Add the private network IP addresses and hostnames of all BMSs in the cluster. For example, run the following commands:

   192.168.0.1 bms-0004

   192.168.0.2 bms-0005

   ...
6. Run the following command to add the **hostfile** file:

   $vi hostfile

   Add hostnames of all BMSs in the cluster, for example:

   bms-0004

   bms-0005

   ...

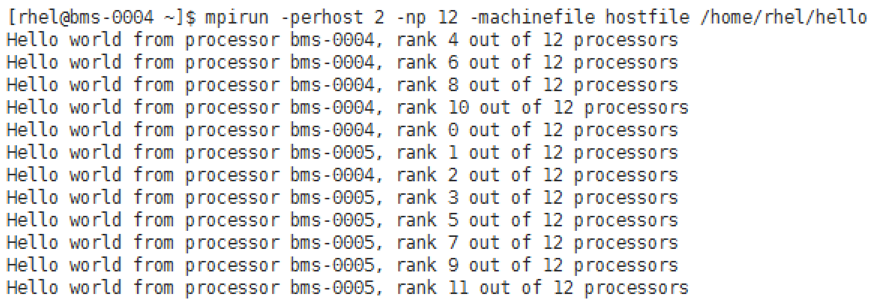
7. Log in to all BMSs in the cluster and repeat **Step 2.5** to **Step 2.6**.

Step 3 Run Intel MPI in the BMS cluster:

For example, there are two BMSs in the cluster. Then, run the following command:

$ mpirun -perhost 2 -np 12 -machinefile hostfile /home/rhel/hello

Figure 4-45 Successful execution of Intel MPI in the BMS cluster



```
[rhel@bms-0004 ~]$ mpirun -perhost 2 -np 12 -machinefile hostfile /home/rhel/hello
Hello world from processor bms-0004, rank 4 out of 12 processors
Hello world from processor bms-0004, rank 6 out of 12 processors
Hello world from processor bms-0004, rank 8 out of 12 processors
Hello world from processor bms-0004, rank 10 out of 12 processors
Hello world from processor bms-0004, rank 0 out of 12 processors
Hello world from processor bms-0005, rank 1 out of 12 processors
Hello world from processor bms-0004, rank 2 out of 12 processors
Hello world from processor bms-0005, rank 3 out of 12 processors
Hello world from processor bms-0005, rank 5 out of 12 processors
Hello world from processor bms-0005, rank 7 out of 12 processors
Hello world from processor bms-0005, rank 9 out of 12 processors
Hello world from processor bms-0005, rank 11 out of 12 processors
```

NOTE

Specify the path of **hostfile** when running it. The path of the executable file **hello** must be absolute. All executable files in the cluster must be in the same directory.

----End

4.5.5 Platform MPI

Scenarios

This section describes how to run Platform MPI (version platform_mpi-09.01.04.03r-ce.bin is used as an example) in a BMS cluster.

Prerequisites

- Password-free login has been configured between BMSs in the cluster.
- Platform MPI has been installed on all BMSs in the cluster.

Procedure

Step 1 Disable the firewall.

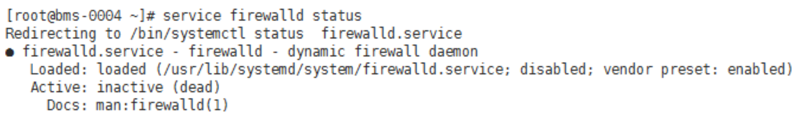
1. Log in to a BMS in the cluster.
2. Run the following command to disable the BMS firewall:

   # service firewalld stop

   # iptables -F
3. Run the following command to check whether the firewall has been disabled:

   # service firewalld status

Figure 4-46 Disabled firewall



```
[root@bms-0004 ~]# service firewalld status
Redirecting to /bin/systemctl status  firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
```

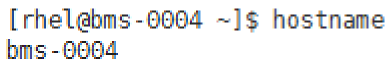
4. Log in to all other BMSs in the cluster and repeat **Step 1.2** to **Step 1.3** to disable firewalls on all BMSs.

Step 2 Modify the configuration file.

1. Log in to a BMS in the cluster.
2. Run the following command to obtain the BMS hostname:

   $ hostname

   Figure 4-47 Viewing the BMS hostname

```
[rhel@bms-0004 ~]$ hostname
bms-0004
```

3. Log in to all other BMSs in the cluster and repeat **Step 2.1** to **Step 2.2** to obtain hostnames of all BMSs.
4. Log in to a BMS in the cluster.
5. Run the following command to add the hosts configuration file:

   # vi /etc/hosts

   Add the private network IP addresses and hostnames of all BMSs in the cluster. For example, run the following commands:

   192.168.0.1 bms-0004

   192.168.0.2 bms-0005

   ...
6. Run the following command to add the **hostfile** file:

   $vi hostfile

   Add hostnames of all BMSs in the cluster, for example:

   bms-0004

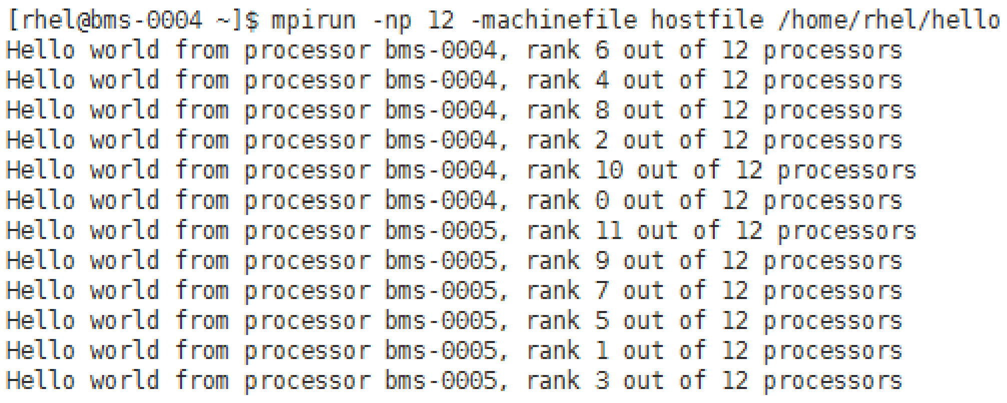
   bms-0005

   ...
7. Log in to all BMSs in the cluster and repeat **Step 2.5** to **Step 2.6**.

Step 3 Run the following command to run Platform Open MPI in the BMS cluster:

$ mpirun -np 12 -machinefile hostfile /home/rhel/hello

Figure 4-48 Successful execution of Platform MPI in the BMS cluster



```
[rhel@bms-0004 ~]$ mpirun -np 12 -machinefile hostfile /home/rhel/hello
Hello world from processor bms-0004, rank 6 out of 12 processors
Hello world from processor bms-0004, rank 4 out of 12 processors
Hello world from processor bms-0004, rank 8 out of 12 processors
Hello world from processor bms-0004, rank 2 out of 12 processors
Hello world from processor bms-0004, rank 10 out of 12 processors
Hello world from processor bms-0004, rank 0 out of 12 processors
Hello world from processor bms-0005, rank 11 out of 12 processors
Hello world from processor bms-0005, rank 9 out of 12 processors
Hello world from processor bms-0005, rank 7 out of 12 processors
Hello world from processor bms-0005, rank 5 out of 12 processors
Hello world from processor bms-0005, rank 1 out of 12 processors
Hello world from processor bms-0005, rank 3 out of 12 processors
```

NOTE

Specify the path of **hostfile** when running it. The path of the executable file **hello** must be absolute. All executable files in the cluster must be in the same directory.

----End

4.6 Running MPI Applications in an HPC Cluster (Kunpeng BMS)

This section uses the CentOS 7.6 OS as an example to describe how to run MPIs in a Kunpeng BMS cluster.

4.6.1 Open MPI Delivered with the IB Driver

Scenarios

This section describes how to run Open MPI (version 4.0.2a1 is used as an example) delivered with the IB driver in a Kunpeng BMS cluster.

Prerequisites

- Password-free login has been configured between BMSs in the cluster.
- Open MPI delivered with the IB driver has been installed on all BMSs in the cluster.

Procedure

Step 1 Disable the firewall.

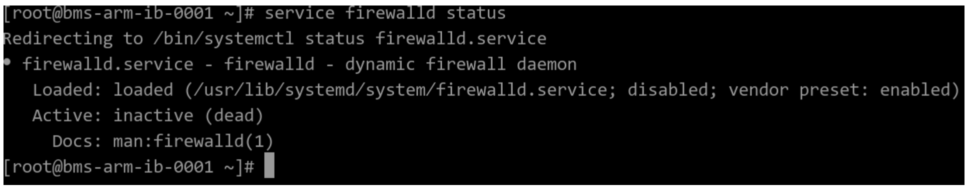
1. Log in to a BMS in the cluster.
2. Run the following commands to disable the BMS firewall:

   # service firewalld stop

   # iptables -F
3. Run the following command to check whether the firewall has been disabled:

   # service firewalld status

   Figure 4-49 Disabled firewall

```
[root@bms-arm-ib-0001 ~]# service firewalld status
Redirecting to /bin/systemctl status firewalld.service
• firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
[root@bms-arm-ib-0001 ~]#
```

4. Log in to all other BMSs in the cluster and repeat **Step 1.2** to **Step 1.3** to disable firewalls on all BMSs.

Step 2 Modify the configuration file.

1. Log in to any BMS in the cluster and run the following command to add the hosts configuration file:

   # vi /etc/hosts

   Add the private network IP addresses and hostnames of all BMSs in the cluster. For example, run the following commands:

   192.168.1.138 bms-arm-ib-0001

   192.168.1.45 bms-arm-ib-0002

   ...

2. Run the following command to add the **hostfile** file:

   $vi hostfile

   Add the hostnames of all BMSs in the cluster and the number of cores (for example, 2 cores).

   bms-arm-ib-0001 slots=2

   bms-arm-ib-0002 slots=2

   ...

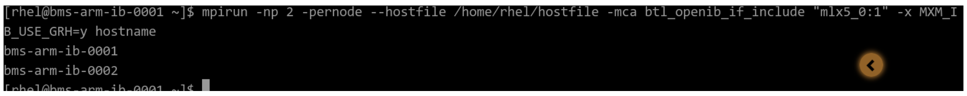
3. Log in to all BMSs in the cluster and repeat **Step 2.1** to **Step 2.3**.

Step 3 Run MPI benchmark.

1. Perform the following command on any BMS to check whether the **hostfile** file has been configured:

   $ mpirun -np 2 -pernode --hostfile hostfile -mca btl_openib_if_include "mlx5_0:1" -x MXM_IB_USE_GRH=y hostname

   Figure 4-50 Checking the configuration file

```
[rhel@bms-arm-ib-0001 ~]$ mpirun -np 2 -pernode --hostfile /home/rhel/hostfile -mca btl_openib_if_include "mlx5_0:1" -x MXM_I
B_USE_GRH=y hostname
bms-arm-ib-0001
bms-arm-ib-0002
[rhel@bms-arm-ib-0001 ~]$ 
```

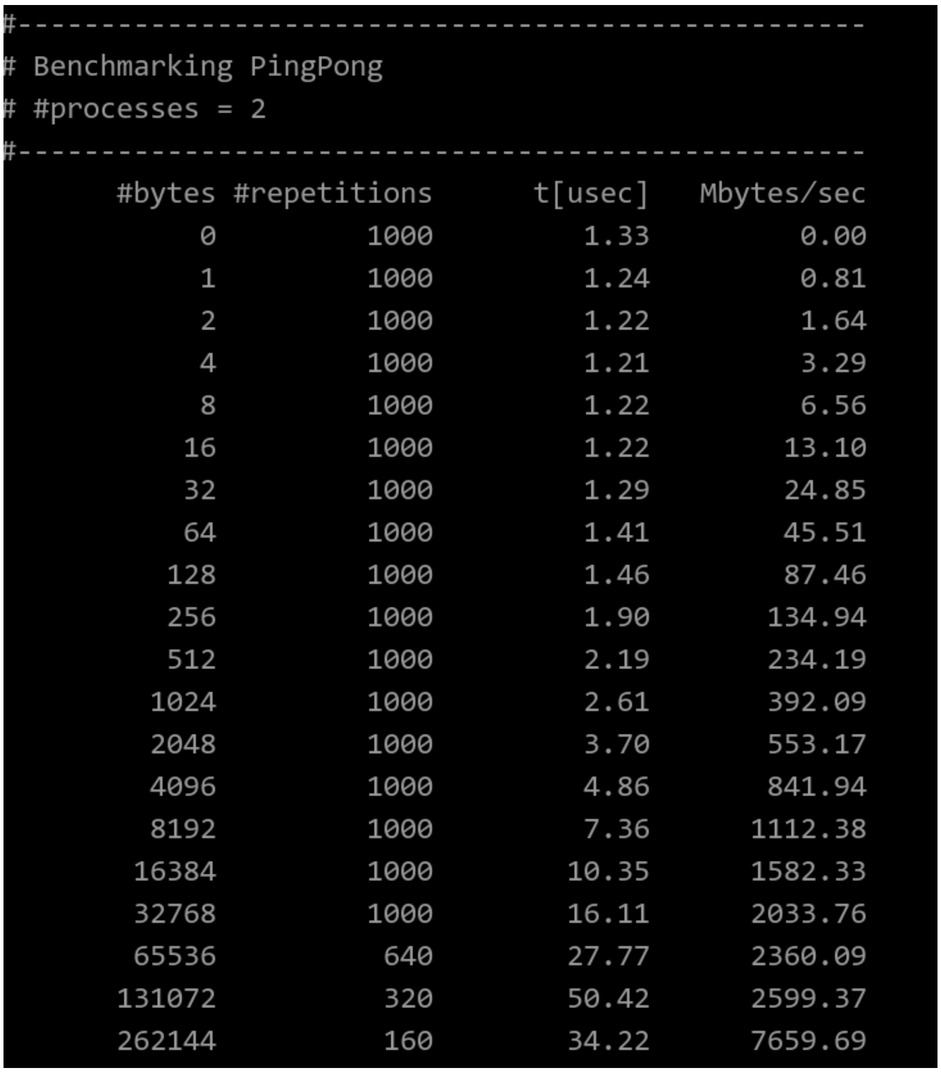
   If the hostnames of all BMSs in the cluster are displayed, as shown in **Figure 4-50**, the **hostfile** file has been configured.

2. Run the MPI benchmark on any BMS with the hostfile path specified.

   For example, there are two BMSs in the cluster. Then, run the following command:

   $ mpirun -np 2 -pernode --hostfile hostfile -mca btl_openib_if_include "mlx5_0:1" -x MXM_IB_USE_GRH=y /usr/mpi/gcc/openmpi-4.0.2a1/tests/imb/IMB-MPI1 PingPong

Figure 4-51 Running Open MPI delivered with the IB driver in the cluster



```
#----------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#----------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec
            0         1000         1.33         0.00
            1         1000         1.24         0.81
            2         1000         1.22         1.64
            4         1000         1.21         3.29
            8         1000         1.22         6.56
           16         1000         1.22        13.10
           32         1000         1.29        24.85
           64         1000         1.41        45.51
          128         1000         1.46        87.46
          256         1000         1.90       134.94
          512         1000         2.19       234.19
         1024         1000         2.61       392.09
         2048         1000         3.70       553.17
         4096         1000         4.86       841.94
         8192         1000         7.36      1112.38
        16384         1000        10.35      1582.33
        32768         1000        16.11      2033.76
        65536          640        27.77      2360.09
       131072          320        50.42      2599.37
       262144          160        34.22      7659.69
```

If information shown in **Figure 4-51** is displayed, Open MPI delivered with the IB driver is running in the cluster.

----End

4.6.2 Community Open MPI

Scenarios

This section describes how to run community Open MPI (version 4.0.2 is used as an example) in a BMS cluster.

Prerequisites

- Password-free login has been configured between BMSs in the cluster.
- Community Open MPI has been installed on all BMSs in the cluster.

Procedure

Step 1 Disable the firewall.

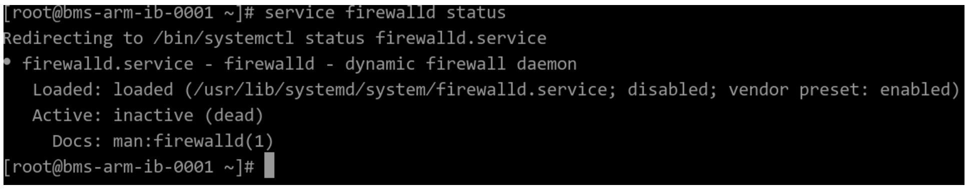
1. Log in to a BMS in the cluster.
2. Run the following commands to disable the BMS firewall:

   # service firewalld stop

   # iptables -F
3. Run the following command to check whether the firewall has been disabled:

   # service firewalld status

   Figure 4-52 Disabled firewall

```
[root@bms-arm-ib-0001 ~]# service firewalld status
Redirecting to /bin/systemctl status firewalld.service
• firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
[root@bms-arm-ib-0001 ~]#
```

4. Log in to all other BMSs in the cluster and repeat **Step 1.2** to **Step 1.3** to disable firewalls on all BMSs.

Step 2 Modify the configuration file.

1. Log in to any BMS in the cluster and run the following command to add the hosts configuration file:

   # vi /etc/hosts

   Add the private network IP addresses and hostnames of all BMSs in the cluster. For example, run the following commands:

   192.168.1.138 bms-arm-ib-0001

   192.168.1.45 bms-arm-ib-0002

   ...
2. Run the following command to add the **hostfile** file:

   $vi hostfile

   Add the hostnames of all BMSs in the cluster and the number of cores (for example, 2 cores).

   bms-arm-ib-0001 slots=2

   bms-arm-ib-0002 slots=2

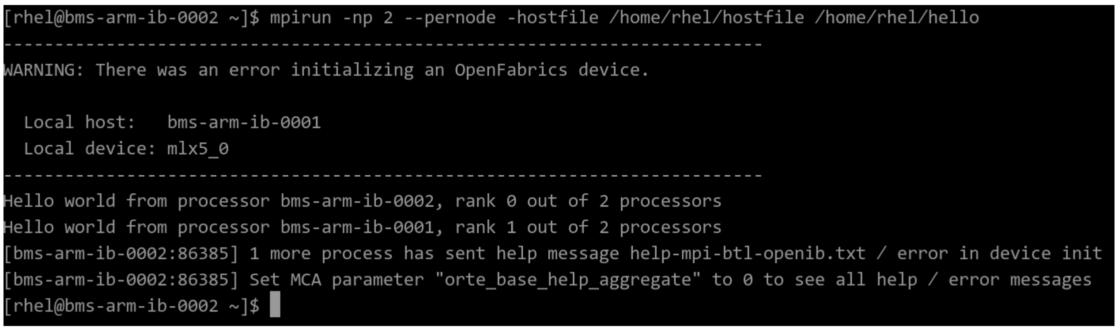
   ...
3. Log in to all BMSs in the cluster and repeat **Step 2.1** to **Step 2.3**.

Step 3 Log in to any BMS and run the community Open MPI.

For example, there are two BMSs in the cluster. Then, run the following command:

$ mpirun -np 2 --pernode -hostfile hostfile /home/rhel/hello

Figure 4-53 Successful execution of community Open MPI in the cluster



```
[rhel@bms-arm-ib-0002 ~]$ mpirun -np 2 --pernode -hostfile /home/rhel/hostfile /home/rhel/hello
--------------------------------------------------------------------------
WARNING: There was an error initializing an OpenFabrics device.

  Local host:   bms-arm-ib-0001
  Local device: mlx5_0
--------------------------------------------------------------------------
Hello world from processor bms-arm-ib-0002, rank 0 out of 2 processors
Hello world from processor bms-arm-ib-0001, rank 1 out of 2 processors
[bms-arm-ib-0002:86385] 1 more process has sent help message help-mpi-btl-openib.txt / error in device init
[bms-arm-ib-0002:86385] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
[rhel@bms-arm-ib-0002 ~]$
```

NOTE

Specify the path of **hostfile** when running it. The path of the executable file **hello** must be absolute. All executable files in the cluster must be in the same directory.

----End

4.6.3 MPICH

Scenarios

This section describes how to run MPICH (version mpich-3.3.2 is used as an example) in a BMS cluster.

Prerequisites

- Password-free login has been configured between BMSs in the cluster.
- MPICH has been installed on all BMSs in the cluster.

Procedure

Step 1 Disable the firewall.

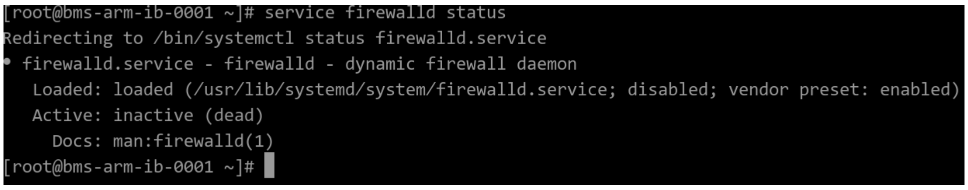
1. Log in to a BMS in the cluster.
2. Run the following commands to disable the BMS firewall:

   # service firewalld stop

   # iptables -F
3. Run the following command to check whether the firewall has been disabled:

   # service firewalld status

   Figure 4-54 Disabled firewall

```
[root@bms-arm-ib-0001 ~]# service firewalld status
Redirecting to /bin/systemctl status firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)
[root@bms-arm-ib-0001 ~]#
```

4. Log in to all other BMSs in the cluster and repeat **Step 1.2** to **Step 1.3** to disable firewalls on all BMSs.

Step 2 Modify the configuration file.

1. Log in to any BMS in the cluster and run the following command to add the hosts configuration file:

   # vi /etc/hosts

   Add the private network IP addresses and hostnames of all BMSs in the cluster. For example, run the following commands:

   192.168.1.138 bms-arm-ib-0001

   192.168.1.45 bms-arm-ib-0002

   ...

2. Run the following command to add the **hostfile** file:

   $vi hostfile

   Add the hostnames of all BMSs in the cluster and the number of cores (for example, 2 cores).

   bms-arm-ib-0001:2

   bms-arm-ib-0002:2

   ...

3. Log in to all BMSs in the cluster and repeat **Step 2.1** to **Step 2.2**.

Step 3 Run the executable file on a BMS through MPICH.

$ mpirun -np 2 -hostfile /home/rhel/hostfile /home/rhel/hello

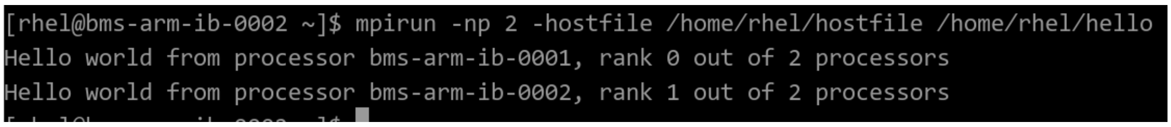


Figure 4-55 Successful execution of MPICH in the BMS cluster

```
[rhel@bms-arm-ib-0002 ~]$ mpirun -np 2 -hostfile /home/rhel/hostfile /home/rhel/hello
Hello world from processor bms-arm-ib-0001, rank 0 out of 2 processors
Hello world from processor bms-arm-ib-0002, rank 1 out of 2 processors
```

NOTE

Specify the path of **hostfile** when running it. The path of the executable file **hello** must be absolute. All executable files in the cluster must be in the same directory.

----End