陕西理工大学
Shaanxi University of Technology

陕西理工 HPC 用户手册

Release 1.0

HPC

Sep 08, 2019

Contents

CHAPTER 1

系统说明

1.1 系统组成

当前系统包含如下组件：

- 管理节点：1 台管理节点
- 登录节点：2 台登录节点
- 计算节点：52 台
- 存储系统：管理节点内置 14*6TB
- 管理网络：4 个千兆交换机
- 计算网络：3 台 36 端口 InfiniBand 交换机

1.2 系统逻辑架构

当前系统架构图：

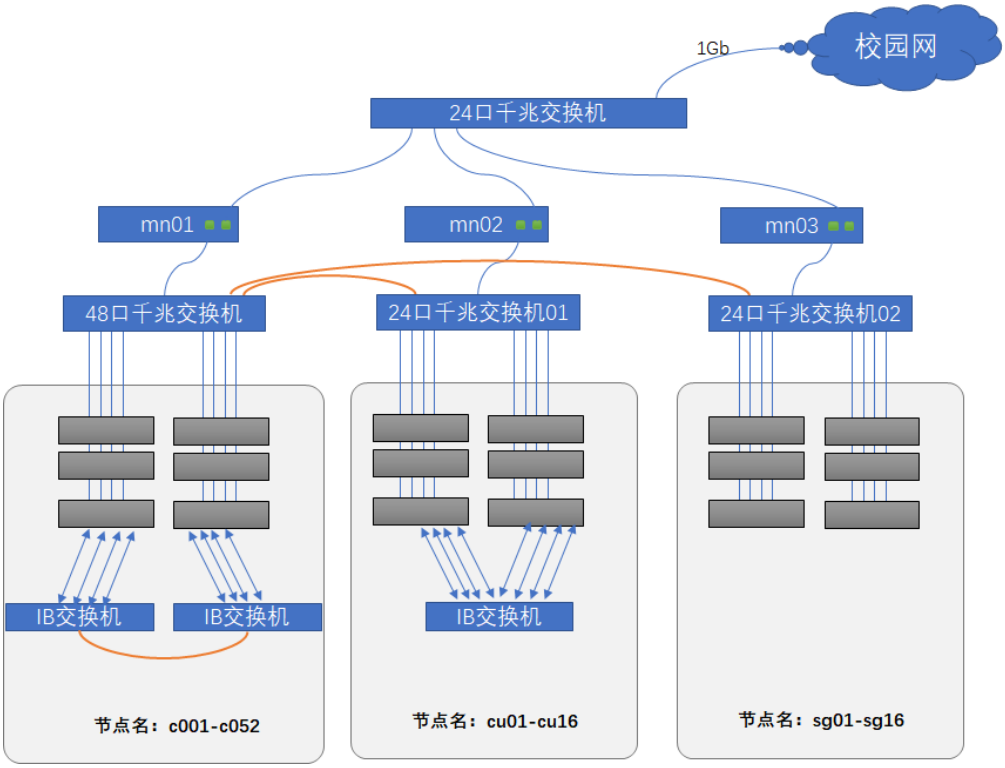


Fig. 1: HPC 系统架构

1.3 系统物理视图

当前系统占用 7 个机柜空间，A 列 3 个机柜，B 列 4 个机柜。

- A 列：新管理节点，3 个 GPU 节点，52 个新计算节点
- B 列：2 个老管理节点，32 个老计算节点

A 列机柜的物理视图如下：

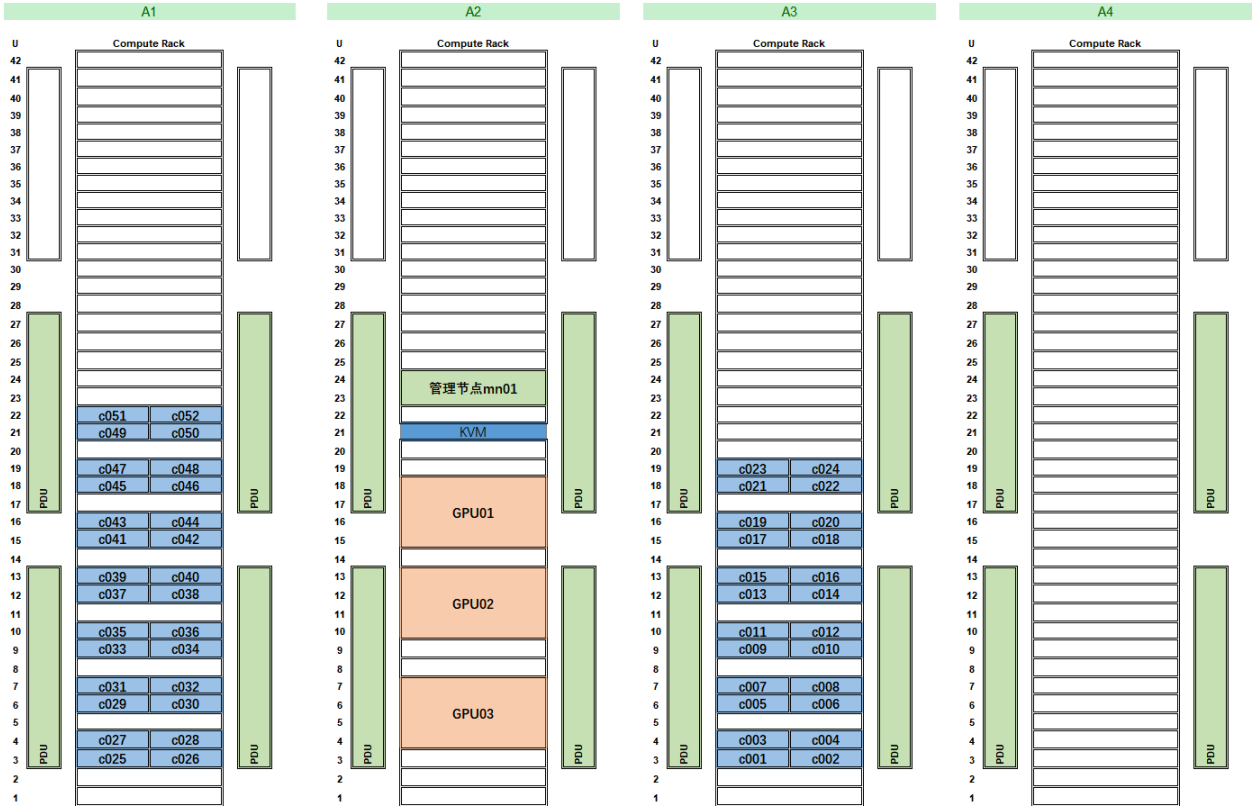


Fig. 2: A 列机柜物理视图

B 列机柜的物理视图如下：

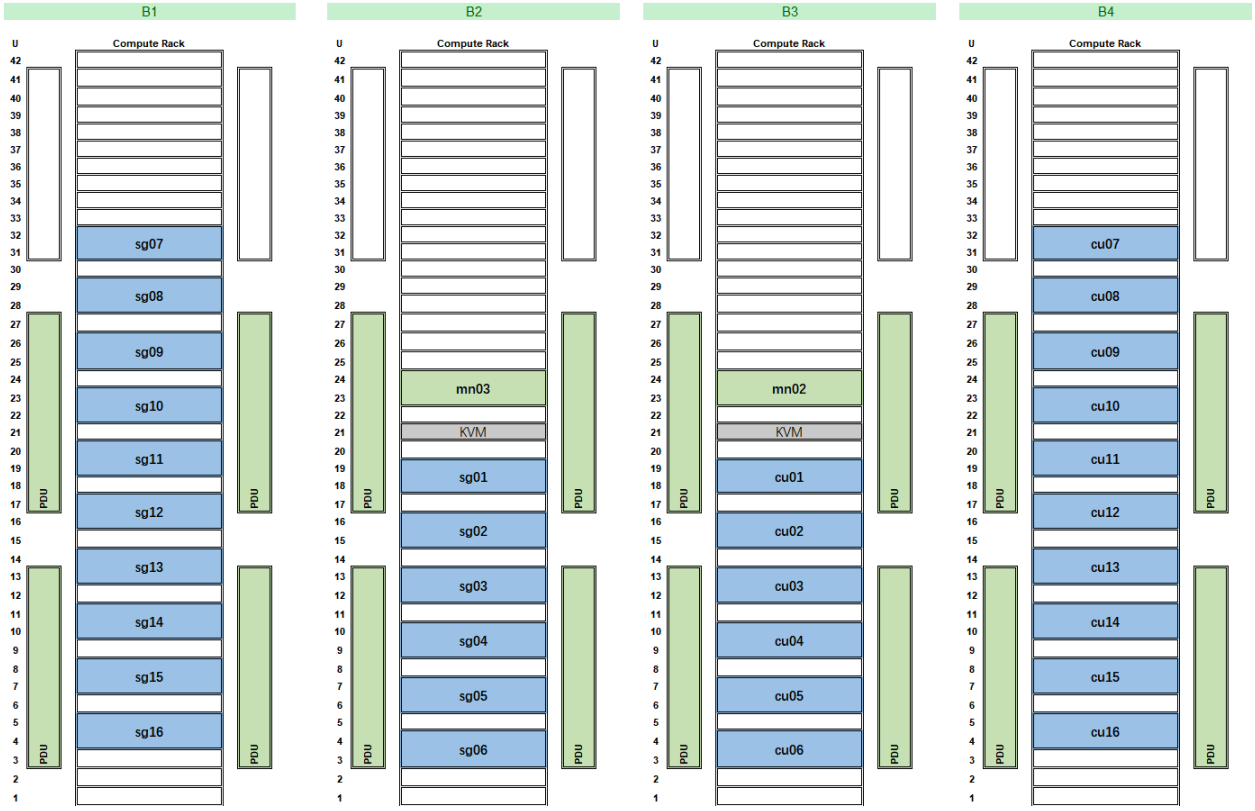


Fig. 3: B 列机柜物理视图

1.4 节点命名及 IP 地址

节点类型	主机名	IP 地址
云海管理节点	mn01	10.1.1.254
云海节点	c001 ~ c052	10.1.1.1 ~ 10.1.1.52
GPU 节点	gpu01 ~ gpu03	10.1.1.201 ~ 10.1.1.203

节点类型	主机名	IP 地址
浪潮管理节点	mn02	10.1.1.253
浪潮节点	cu01 ~ cu16	10.1.1.101 ~ 10.1.1.116

节点类型	主机名	IP 地址
曙光管理节点	mn03	10.1.1.252
曙光节点	sg01 ~ sg16	10.1.1.121 ~ 10.1.1.136

1.5 集群软件

当前集群软件系统相关描述如下：

Table 1: 集群软件信息

名称	版本
操作系统	CentOS Linux release 7.6.1810
Intel	Intel Parallel Studio XE 2017/2018/2019
编译器	GCC 4.8.5(系统自带), 7.4, 8.3, 9.1
数学库	Intel MKL 2017/2018/2019
并行环境	Intel MPI 2017/2018/2019, OpenMPI 3.1.4/4.0.1
作业调度	PBSPro 19.1.1
共享存储	NFS v4
集群管理	xCAT xcat-2.14.5

Note: 请勿使用 yum/rpm 更新系统重要软件包，如 kernel、glibc 等，更新将有可能导致集群系统无法正常工作。

Warning: 禁止在节点上安装和更新系统 yum 源之外的软件包。

CHAPTER 2

系统管理

2.1 管理节点 RAID 管理

2.1.1 存储概述

当前系统的共享存储由管理节点提供。

当前 RAID 卡共接 16 个硬盘，其中 2 个 900GB，做 RAID1，安装操作系统。磁盘位置为 slot0,slot1。

余下 14 个 6TB，其中 slot2 ~ slot14 设置为 RAID6，slot15 设置为热备盘。

RAID 后的磁盘容量为：60TiB

为保证数据安全性，需定时查看硬盘状态。处于最优状态。

- 在机房本地查看硬盘灯状态。
- 在远程用 MegaRAID Storage Manager 查看 RAID 状态。

2.1.2 RAID 卡管理软件

RAID 卡管理界面启动命令：

```
/usr/local/MegaRAID\ Storage\ Manager/startupui.sh
```

MegaRAID Storage Manager 界面：

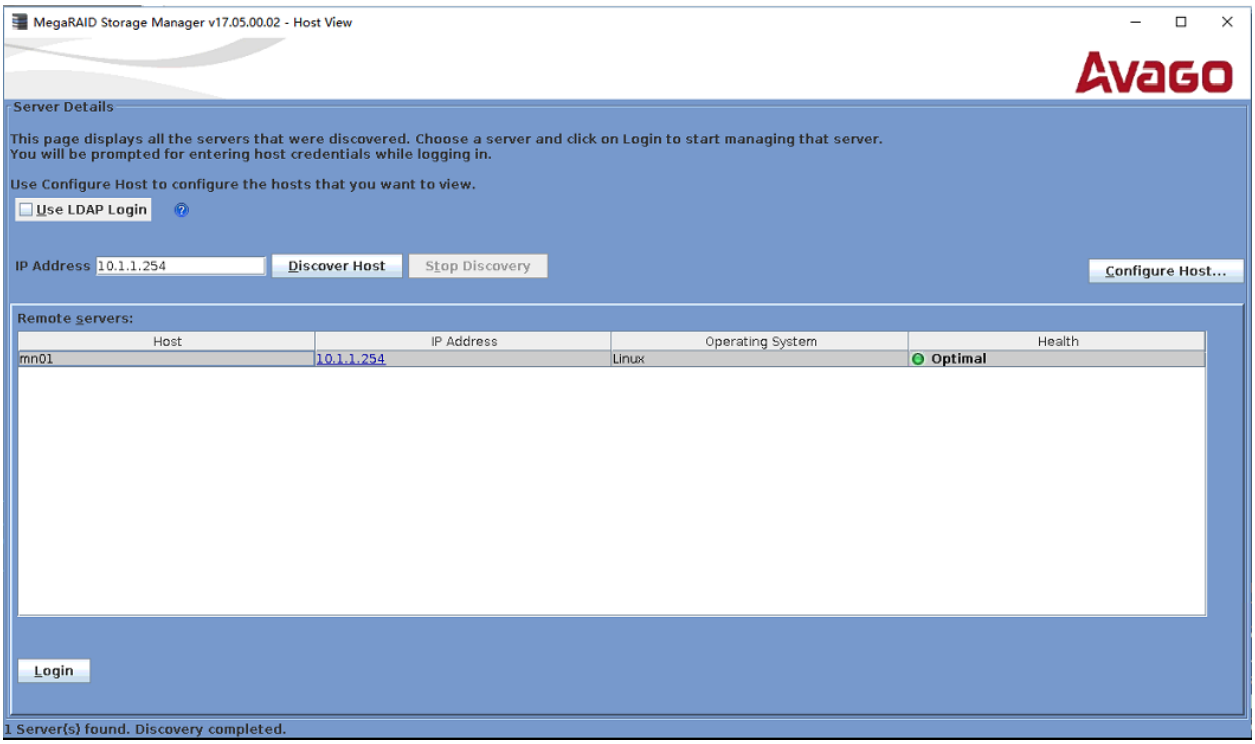


Fig. 1: MegaRAID Storage Manager 初始界面

在此界面可以双击主机进入管理界面，账户密码为当前主机的管理员账户和密码。

Fig. 2: MegaRAID Storage Manager 密码界面

MSM 登录后：

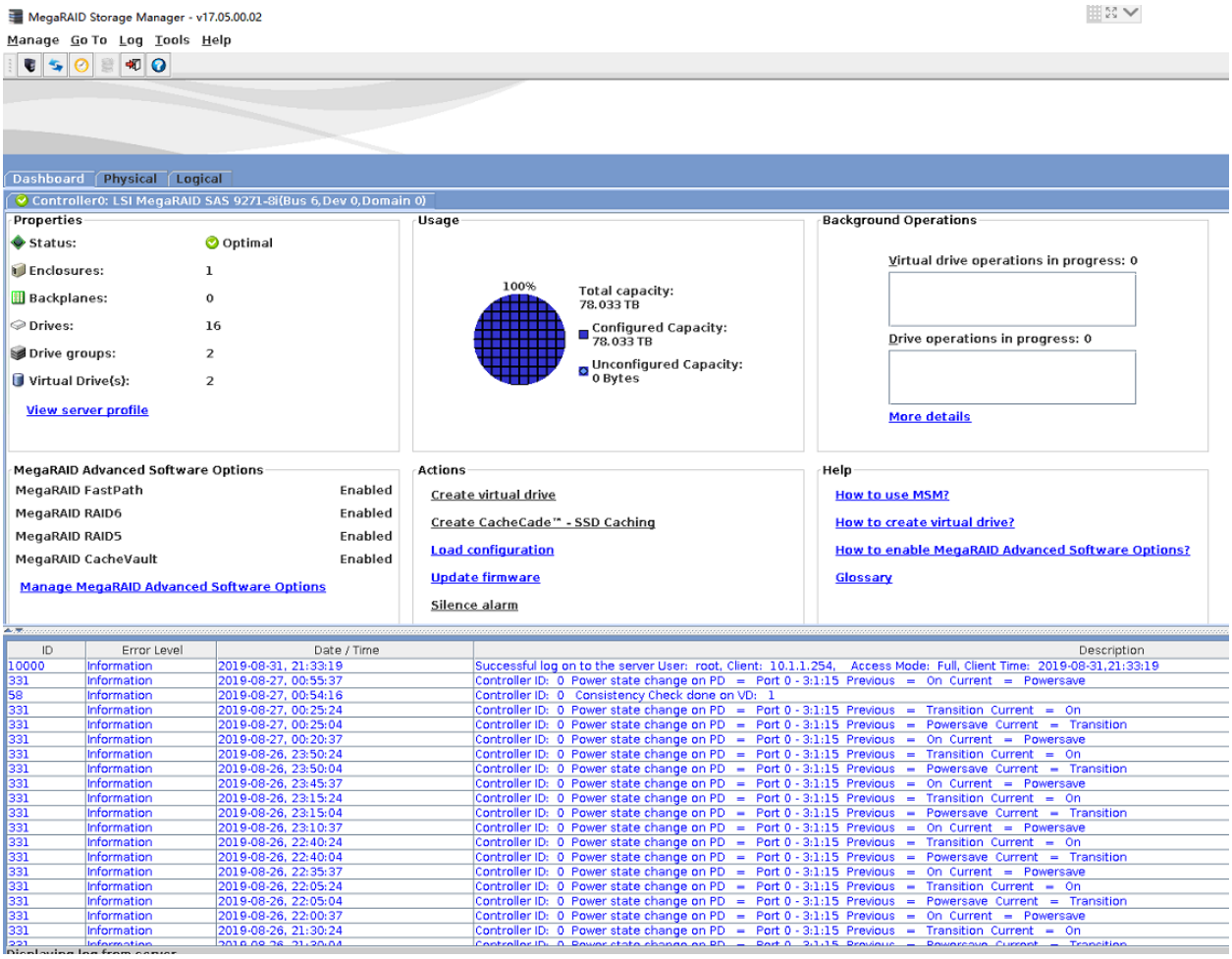
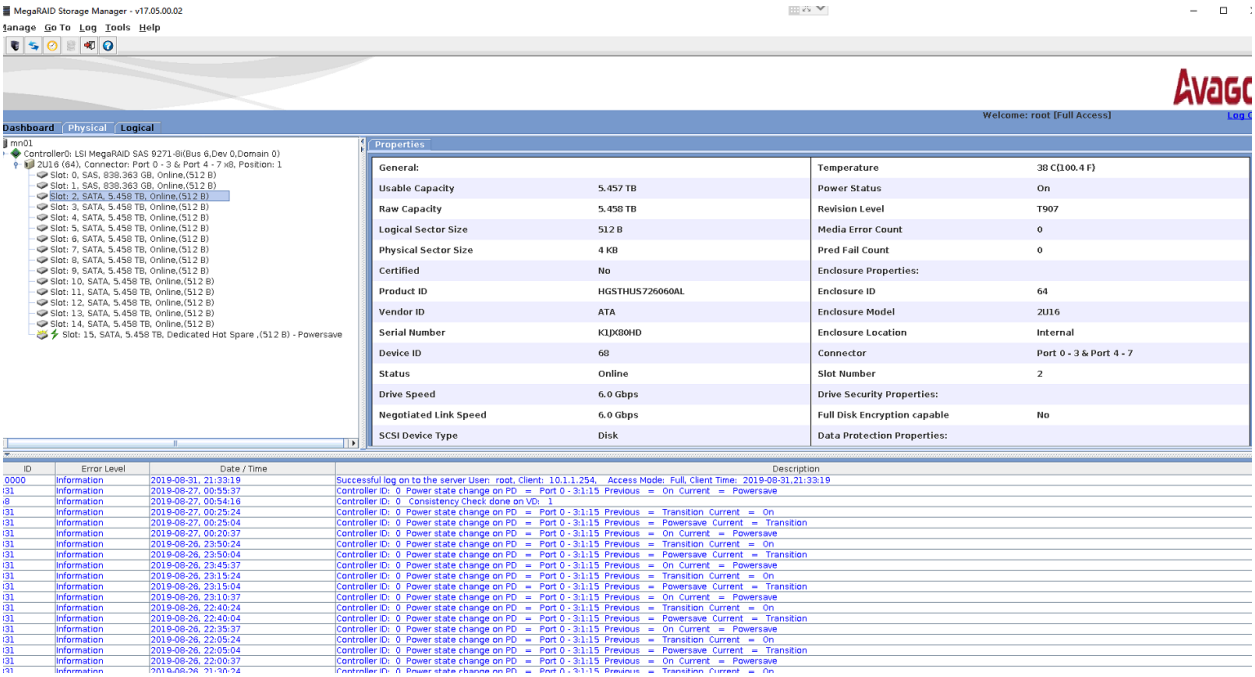


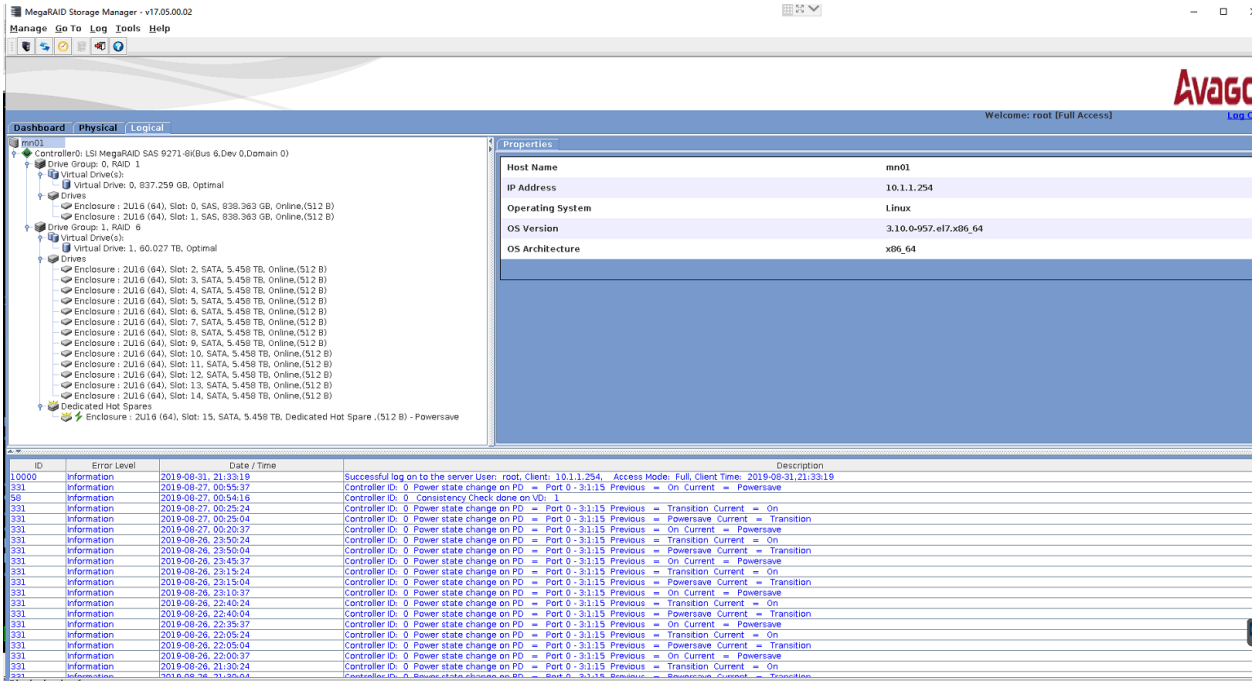
Fig. 3: MegaRAID Storage Manager 管理主界面

在 Dashboard 界面中 RAID 的状态应该处于 `Optimal` 状态

2.1.3 查看物理硬盘状态



2.1.4 查看逻辑卷状态



2.2 集群系统基础服务

同时，系统的 DNS、DHCP、TFTP、NTP、HTTP 服务，均由 xCAT 管理。

> **Warning:** 请勿手动修改系统的 DNS、DHCP、TFTP 服务。

日常使用时，需要保证 NTP 服务工作正常，其他 DNS、DHCP、TFTP、HTTP 仅在系统部署时需要保证运行正常。

查看 NTP 服务运行状态：

```
[root@mn01 ~]# systemctl status chronyd
● chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor preset:●
↪enabled)
   Active: active (running) since Wed 2019-06-26 22:14:45 CST; 6 days ago
     Docs: man:chronyd(8)
           man:chrony.conf(5)
 Main PID: 186465 (chronyd)
    Tasks: 1
   CGroup: /system.slice/chronyd.service
           ●●186465 /usr/sbin/chronyd
```

2.2.1 节点状态

在系统中，提供了命令行查看节点状态的命令，分别为 whatsup 和 nodestat 。

whatsup

whatsup 通过定时 ping 节点来判断节点的状态。

```
[root@mn01 ~]# whatsup
up:   85: c[001-052],cu[01-16],gpu[01-03],sg[01-06,08,10-16]
down:  2: sg[07,09]
[root@mn01 ~]#
```

nodestat

nodestat 是 xCAT 提供的查看节点状态的命令，不同于 whatsup ，nodestat 能确认节点是否处于正常的操作系统运行状态下，能判断节点是否能 ssh 或者运行 pbs。

```
[root@mn01 ~]# nodestat  c[001-010]
c001: pbs,sshd
c002: pbs,sshd
c003: pbs,sshd
c004: pbs,sshd
c005: pbs,sshd
c006: pbs,sshd
c007: pbs,sshd
```

(continues on next page)

(continued from previous page)

```
c008: pbs,sshd
c009: pbs,sshd
c010: pbs,sshd
[root@mn01 ~]#
```

nodestat 命令依赖于 xcatd 服务。需确认 xcatd 服务正常运行，命令如下：

```
[root@mn01 ~]# systemctl status xcatd
● xcatd.service - xCAT management service
   Loaded: loaded (/usr/lib/systemd/system/xcatd.service; enabled; vendor preset:●
↪disabled)
   Active: active (running) since Sun 2019-08-04 10:37:20 CST; 3 weeks 6 days ago
 Main PID: 19606 (xcatd: SSL list)
    Tasks: 7
   Memory: 58.4M
   CGroup: /system.slice/xcatd.service
           ├─19605 /usr/sbin/in.tftpd -v -l -s /tftpboot -m /etc/tftpmapfile4xcat.conf
           ├─19606 xcatd: SSL listener
           ├─19607 xcatd: DB Access
           ├─19608 xcatd: UDP listener
           ├─19609 xcatd: install monitor
           ├─19610 xcatd: Discovery worker
           └─19611 xcatd: Command log writer
```

2.3 集群账户管理

2.3.1 NIS 部署

集群采用 NIS 进行账户数据管理，mn01 设置为 NIS 的 Master。默认的，NIS 服务随 mn01 节点启动。

2.3.2 NIS 服务启停

启动 NIS 服务

```
[root@mn01 ~]# systemctl start ypserv
[root@mn01 ~]# systemctl start yppasswdd
```

停止 NIS 服务

```
[root@mn01 ~]# systemctl stop ypserv
[root@mn01 ~]# systemctl stop yppasswdd
```

2.3.3 NIS 服务状态

在 mn01 上确保服下服务正常运行：

- ypserv

```
[root@mn01 ~]# systemctl status ypserv
● ypserv.service - NIS/YP (Network Information Service) Server
   Loaded: loaded (/usr/lib/systemd/system/ypserv.service; enabled; vendor preset:●
↪disabled)
   Active: active (running) since Sun 2019-01-06 16:20:53 CST; 3 weeks 0 days ago
 Main PID: 216182 (ypserv)
   Status: "Processing requests..."
    Tasks: 1
   CGroup: /system.slice/ypserv.service
           ●●216182 /usr/sbin/ypserv -f

Jan 06 16:20:53 mn01 systemd[1]: Starting NIS/YP (Network Information Service) Server.
↪..
Jan 06 16:20:53 mn01 ypserv[216182]: WARNING: no securenets file found!
Jan 06 16:20:53 mn01 systemd[1]: Started NIS/YP (Network Information Service) Server.
```

- yppasswdd

```
[root@mn01 ~]# systemctl status yppasswdd
● yppasswdd.service - NIS/YP (Network Information Service) Users Passwords Change●
↪Server
   Loaded: loaded (/usr/lib/systemd/system/yppasswdd.service; enabled; vendor preset:●
↪disabled)
   Active: active (running) since Sun 2019-01-06 16:20:49 CST; 3 weeks 0 days ago
 Main PID: 216163 (rpc.yppasswdd)
   Status: "Processing requests..."
    Tasks: 1
   CGroup: /system.slice/yppasswdd.service
           ●●216163 /usr/sbin/rpc.yppasswdd -f

Jan 06 16:20:49 mn01 systemd[1]: Starting NIS/YP (Network Information Service) Users●
↪Passwords Change Server...
Jan 06 16:20:49 mn01 systemd[1]: Started NIS/YP (Network Information Service) Users●
↪Passwords Change Server.
```

2.3.4 创建删除用户

集群账户的创建和修改，与通常的 Linux 账户管理一致，唯一差异是，需要在进行账户管理后，进行 NIS 数据库更新。

添加账户：useradd <username>

删除账户：userdel <username>

修改账户：usermod <username>

更新数据：make -C /var/yp

Note: 账户管理仅在 mn01 上进行，禁止在 mn02 上管理。Master 信息会被自动传送到 Slave。

确认节点账户信息

使用 `id` 命令确认节点账户信息

```
[root@mn01 images]# psh compute id test
    c001: uid=1000(test) gid=1000(test) groups=1000(test)
    c002: uid=1000(test) gid=1000(test) groups=1000(test)
    c003: uid=1000(test) gid=1000(test) groups=1000(test)
    c004: uid=1000(test) gid=1000(test) groups=1000(test)
```

2.4 NFS 管理

系统采用 NFS 提供文件共享服务。需要确保管理节点 nfs 服务正常运行，计算节点 autofs 服务正常运行。

2.4.1 管理节点 NFS 服务

NFS 服务启动及停止

```
[root@mn01 ~]# systemctl start nfs
[root@mn01 ~]# systemctl stop nfs
```

NFS 服务状态

```
[root@mn01 ~]# systemctl status nfs
    nfs-server.service - NFS server and services
       Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled; vendor
↪preset: disabled)
      Drop-In: /run/systemd/generator/nfs-server.service.d
               order-with-mounts.conf
       Active: active (exited) since Wed 2019-07-03 19:03:25 CST; 1h 57min ago
     Main PID: 432537 (code=exited, status=0/SUCCESS)
        Tasks: 0
       CGroup: /system.slice/nfs-server.service

    Jul 03 19:03:25 mn01 systemd[1]: Starting NFS server and services...
    Jul 03 19:03:25 mn01 systemd[1]: Started NFS server and services.
```

2.4.2 计算节点 autofs 服务状态

默认的，节点正常启动后，autofs 服务会自动启动。

autofs 服务启动及停止

```
[root@mn01 ~]# systemctl start autofs
[root@mn01 ~]# systemctl stop autofs
```

autofs 服务状态

```
【root@c001 ~]# systemctl status autofs
    autofs.service - Automounts filesystems on demand
       Loaded: loaded (/usr/lib/systemd/system/autofs.service; enabled; vendor
↪preset: disabled)
       Active: active (running) since Wed 2019-06-26 23:31:45 CST; 6 days ago
     Main PID: 2254 (automount)
```

(continues on next page)

(continued from previous page)

```
        Tasks: 4
      CGroup: /system.slice/autofs.service
              □□2254 /usr/sbin/automount ^^pid-file /run/autofs.pid

   Jun 26 23:31:45 c001.hpc.wedge systemd[1]: Starting Automounts filesystems on□
↪demand...
   Jun 26 23:31:45 c001.hpc.wedge systemd[1]: Started Automounts filesystems on□
↪demand.
```

2.4.3 测试 NFS 文件共享服务

通过在计算节点访问共享目录，确认文件服务正常：

```
[root@mn01 ~]# psh compute ls -l /home | xcoll
    ====================================
    compute
    ====================================
    total 4
    drwx^^^^^^ 19 test                test 4096 Jul  3 19:04 test

    [root@mn01 ~]# psh compute ls -l /share | xcoll
    ====================================
    compute
    ====================================
    total 0
    drwxr-xr-x 4 root root 33 Jul  3 19:40 0.download
    drwxr-xr-x 4 root root 45 Jun 28 15:52 apps
    drwxr-xr-x 3 root root 20 Jun 21 15:33 base
    drwxr-xr-x 3 root root 23 Jul  3 19:08 bench
    drwxr-xr-x 2 root root  6 Jun 26 22:15 intel
    drwxr-xr-x 7 root root 70 Jun 27 00:10 lsfsuites
    drwxrwxrwx 2 root root  6 Jun 28 16:11 temp
```

`ls -l /home` 或者 `ls -l /share` 应能正常列出 **/home** 和 **/share** 目录下的目录或者文件。

使用 “df”命令，确认节点的 **/home** 和 **/share** 目录都是由管理节点提供。

```
[root@mn01 ~]# psh compute 'df -h | grep -E "home|share"' | xcoll
====================================
compute
====================================
mn01:/share          1.1T   60G  1.1T   6% /share
mn01:/home           5.8T   32G  5.8T   1% /home
```

2.5 InfiniBand 管理

系统采用 InfiniBand 提供高速网络服务。需确认如下 InfiniBand 服务正常运行：

2.5.1 InfiniBand 驱动服务

默认的，所有节点（包含管理节点和计算节点）的 openibd 服务会自动开启，使用如下命令确认：

```
[root@mn01 ~]# systemctl status openibd
● openibd.service - openibd - configure Mellanox devices
   Loaded: loaded (/usr/lib/systemd/system/openibd.service; enabled; vendor
↪preset: disabled)
   Active: active (exited) since Mon 2019-06-24 19:21:43 CST; 1 weeks 2 days ago
     Docs: file:/etc/infiniband/openib.conf
 Main PID: 1382 (code=exited, status=0/SUCCESS)
    Tasks: 0
   CGroup: /system.slice/openibd.service

Jun 24 19:21:39 mn01 systemd[1]: Starting openibd - configure Mellanox devices...
Jun 24 19:21:42 mn01 openibd[1382]: Loading HCA driver and Access Layer:[  OK  ]
Jun 24 19:21:43 mn01 systemd[1]: Started openibd - configure Mellanox devices.
```

如 openibd 服务没有启动，可使用如下命令启动：

```
[root@mn01 ~]# systemctl start openibd
```

停止 openibd 服务：

```
[root@mn01 ~]# systemctl stop openibd
```

重启 openibd 服务：

```
[root@mn01 ~]# systemctl restart openibd
```

2.5.2 子网管理器 OpenSM

InfiniBand 依赖 opensm 服务，管理节点默认运行 opensm 服务，使用如下命令确认：

```
[root@mn01 ~]# systemctl status opensmd
● opensmd.service - LSB: Activates/Deactivates InfiniBand Subnet Manager
   Loaded: loaded (/etc/rc.d/init.d/opensmd; bad; vendor preset: disabled)
   Active: active (running) since Mon 2019-06-24 19:21:50 CST; 1 weeks 2 days ago
     Docs: man:systemd-sysv-generator(8)
 Main PID: 2657 (opensm)
    Tasks: 107
   CGroup: /system.slice/opensmd.service
           ●●2657 /usr/sbin/opensm ^^daemon

Jun 24 19:21:49 mn01 systemd[1]: Starting LSB: Activates/Deactivates InfiniBand
↪Subnet Manager...
Jun 24 19:21:49 mn01 OpenSM[2657]: /var/log/opensm.log log file opened
Jun 24 19:21:49 mn01 OpenSM[2657]: OpenSM 5.3.0.MLNX20181108.33944a2
Jun 24 19:21:49 mn01 OpenSM[2657]: Entering DISCOVERING state
Jun 24 19:21:49 mn01 OpenSM[2657]: Entering MASTER state
Jun 24 19:21:50 mn01 opensmd[2615]: Starting IB Subnet Manager.[  OK  ]
Jun 24 19:21:50 mn01 systemd[1]: Started LSB: Activates/Deactivates InfiniBand Subnet
↪Manager.
```

2.5.3 确认 IB 卡状态

使用 ibnodes 命令确认网络内 ib 节点：

```
[root@mn01 ~]# ibnodes
Ca      : 0xb8599f030085c980 ports 2 "c003 HCA-1"
Ca      : 0xb8599f030085ca70 ports 2 "c004 HCA-1"
Ca      : 0xb8599f030085ca50 ports 2 "c001 HCA-1"
Ca      : 0xb8599f030085c970 ports 2 "c002 HCA-1"
Ca      : 0x98039b0300c40eb6 ports 1 "mn01 HCA-1"
Switch  : 0xb8599f03008fe3f0 ports 32 "MF0;switch-0837c6:SX90Y3245/U1" enhanced port
↪0 lid 2 lmc 0
```

节点间，可使用 ping 命令来确认 ib 卡之间的互通情况，例如从管理节点 ping 10.2.1.<#> 。

CHAPTER 3

编译器

3.1 编译器命令

源文件	GCC 命令	Intel 编译器命令
C	gcc	icc
C++	g++	icpc
Fortran	gfortran	ifort

3.2 GCC 编译器

当前系统的 GCC 编译器有 4 个版本，分别为：

版本	路径
gcc 4.8.5	系统默认
7.4	/share/base/gcc/7.4
8.3	/share/base/gcc/8.3
9.1	/share/base/gcc/9.1

3.3 Intel 编译器

Intel C/C++ Fortran 编译器是一种主要针对 Intel 处理器的高性能编译器。

3.3.1 Intel 编译器版本

当前系统安装了 3 个版本的 Intel 编译器，分别为：

- Intel parallel studio XE 2017 update8
- Intel parallel studio XE 2018 update4
- Intel parallel studio XE 2019 update4

3.3.2 Intel 编译器路径

3 个版本编译器分别安装在:

版本	路径
2017 update 8	/share/intel/2017u8
2018 update 4	/share/intel/2018u4
2019 update 4	/share/intel/2019u4

3.3.3 Intel 集群工具在线文档

有关 Intel 编译器、数学库、MPI 和其他组件的完整的文档，请访问：Get Started with Intel Parallel Studio XE[1].

3.3.4 调用方法

系统提供了两种调用方法，分别是 `source` 和 `module load` 。

`source` 方法调用各种 intel 编译器版本方法如下：

- Intel 2017u8: `source /share/intel/2017u8/bin/compilervars.sh intel64`
- Intel 2018u4: `source /share/intel/2018u4/bin/compilervars.sh intel64`
- Intel 2019u4: `source /share/intel/2019u4/bin/compilervars.sh intel64`

确认环境变量加载正确：

```
[test@mn01 ~]$ which icc
/share/intel/2017u8/compilers_and_libraries_2017.8.262/linux/bin/intel64/icc
[test@mn01 ~]$ icc -v
icc version 17.0.8 (gcc version 4.8.5 compatibility)
```

```
[test@mn01 ~]$ which ifort
/share/intel/2017u8/compilers_and_libraries_2017.8.262/linux/bin/intel64/ifort
[test@mn01 ~]$ ifort -v
ifort version 17.0.8
```

[1] https://software.intel.com/en-us/parallel-studio-xe/documentation/get-started

```
[test@mn01 ~]$ which icpc
/share/intel/2017u8/compilers_and_libraries_2017.8.262/linux/bin/intel64/icpc
[test@mn01 ~]$ icpc -v
icpc version 17.0.8 (gcc version 4.8.5 compatibility)
```

module 方法调用：

```
module load intel/2017.8
module load intel/2018.4
module load intel/2019.4
```

有关 module 使用方法，请参考相关章节。

CHAPTER 4

Environment Modules

4.1 Module 简介

在 Linux 集群上，通常会安装有不同版本的多种编译器和其他软件等，如常用的编译器有 intel 和 gnu，常用的 MPI 并行库包括 intel mpi，openmpi，mpich2 等，而且对于同一软件，还包含不同的版本或采用不同编译设置得到的可执行程序和链接库等。在使用这些程序时，经常需要对环境变量进行修改。并且由于程序编译时会调用不同类型编译器或第三库，这时程序之间还存在着依赖关系。这使得当执行某个特定版本的程序时，环境变量的修改变得十分复杂。

Environment Modules 包是一个简化 shell 初始化的工具，它允许用户在使用 modulefiles 进行会话期间轻松修改其环境。每个模块文件都包含为应用程序配置 shell 所需的信息。模块文件可以由系统上的许多用户共享，并且用户可以拥有自己的集合来补充或替换共享模块文件。

4.2 modulefiles 路径

当前系统的 modulefiles 存放在/share/base/modulefiles。

4.3 Modules 使用

module 安装好后，即可调用 module [command] 来查看或加载模块，主要指令如下:

- `module load` 用于加载环境变量文件
- `module list` 用于列出当前已经加载的环境文件
- `module unload` 用于卸载变量文件
- `module avail` 用于查询当前系统已经设置好的变量文件
- `module purge` 用于清楚当前已经加载的所有变量文件

更多关于 module 的信息，请参考 `man module` , `man modulefile`

4.4 Module 例子

查看当前系统 modulefiles：

```
module avail
module av
---------------------------- /share/base/modulefiles/compilers ----------------------
↪-----
gcc/7.4.0          gcc/8.3.0          gcc/9.1.0(default) intel/2017.8       intel/
↪2018.4       intel/2019.4

---------------------------- /share/base/modulefiles/parallel -----------------------
↪----
mpi/intel/2017.8            mpi/intel/2018.4            mpi/intel/2019.4           
↪mpi/openmpi/3.1.4_gcc485    mpi/openmpi/3.1.4_icc2019.4

---------------------------- /share/base/modulefiles/chem ---------------------------
gromacs/4.6.7-intel2019.4-dp gromacs/4.6.7-intel2019.4-sp vasp/5.4.4

---------------------------- /share/base/modulefiles/libraries ----------------------
↪-----
boost/1.70.0  fftw/3.3.4-dp fftw/3.3.4-sp

---------------------------- /share/base/modulefiles/tools --------------------------
↪-
cmake/3.15.2

---------------------------- /usr/share/Modules/modulefiles -------------------------
↪--
dot         module-git  module-info modules     null        use.own
```

加载 intel 编译器：

```
module load intel/2019.4
```

加载 intel 编译器：

```
module load mpi/intel/2019.4
```

加载 vasp 程序：

```
module load vasp/5.4.4
```

加载 gromacs 程序：

```
module load gromacs/4.6.7-intel2019.4-sp
module load gromacs/4.6.7-intel2019.4-dp
```

列出当前加载的变量模块：

```
module list
```

卸载 intel 编译器：

```
module unload intel/2019.4
module rm intel/2019.4
```

清除当前环境中所有变量模块：

```
module purge
```

CHAPTER 5

MPI

5.1 OpenMPI

5.1.1 OpenMPI 版本及路径

系统安装了 2 个版本的 OpenMPI，路径为：/share/apps/openmpi

版本	关联的编译器	路径
3.1.4	gcc 4.8.5	/share/apps/openmpi/3.1.4_gcc485
	intel 2019 update4	/share/apps/openmpi/3.1.4_icc2019.4
4.0.1	gcc 4.8.5	/share/apps/openmpi/4.0.1_gcc485
	intel 2019 update4	/share/apps/openmpi/4.0.1_icc2019.4

5.1.2 modulefiles

可以通过 `module` 命令查看当前系统的 OpenMPI 版本：

```
module avail mpi/openmpi

---------------------- /share/base/modulefiles/parallel ----------------------
mpi/openmpi/3.1.4_gcc485    mpi/openmpi/3.1.4_icc2019.4
```

加载 OpenMPI 环境变量:

```
module load mpi/openmpi/3.1.4_gcc485
```

5.2 Intel MPI

5.2.1 Intel MPI 版本及路径

Intel MPI 包含在 Intel parallel studio XE，系统三个版本安装路径如下：

版本	安装路径
2017.4.262	/share/intel/2017u8/impi/2017.4.262/intel64
2018.4.274	/share/intel/2018u4/impi/2018.4.274/intel64
2019.4.243	/share/intel/2019u4/impi/2019.4.243/intel64

使用 source 命令加载相应版本的 Intel MPI：

```
source /share/intel/2017u8/impi/2017.4.262/intel64/bin/mpivars.sh
source /share/intel/2018u4/impi/2018.4.274/intel64/bin/mpivars.sh
source /share/intel/2019u4/impi/2019.4.243/intel64/bin/mpivars.sh
```

通过 module 命令加载：

```
module load mpi/intel/2017.8
module load mpi/intel/2018.4
module load mpi/intel/2019.4
```

查看系统 Intel MPI modulefiles：

```
module avail mpi/intel

---------------------- /share/base/modulefiles/parallel ----------------------
mpi/intel/2017.8 mpi/intel/2018.4 mpi/intel/2019.4
```

Intel MPI 命令用法及详细信息，请访问在线文档：get started with mpi[2]

[2] https://software.intel.com/en-us/get-started-with-mpi-for-linux

CHAPTER 6

PBS 作业调度系统

6.1 PBS 简介

当前系统采用 PBSPro 进行作业调度管理 (我们简称为 PBS 系统)，PBSPro 是 Altair 的最新的开源版本，部分的命令与 OpenPBS 相兼容，用户需要登录集群管理节点进行作业提交和修改。

在 PBS 系统中，用户使用 qsub 命令提交用户程序。

6.2 队列

当前系统设置了 2 个队列，分别是：

Queue name	nodes	ACL	acl_users
workq	c001-c054	Disable	
smp	s001-s002	Disable	

用户通过 `qsub -q queuename` 指明所提交的作业进入哪一个队列。

默认的，不指明队列名时，作业不会获得节点资源，无法运行。

所有提交的作业，进入系统排队，当系统空闲资源能满足作业所申请的资源时，所提交作业会被立刻安排计算，无足够空闲资源时，作业进入排队状态。

6.3 提交作业

运行 PBS 作业的流程如下：

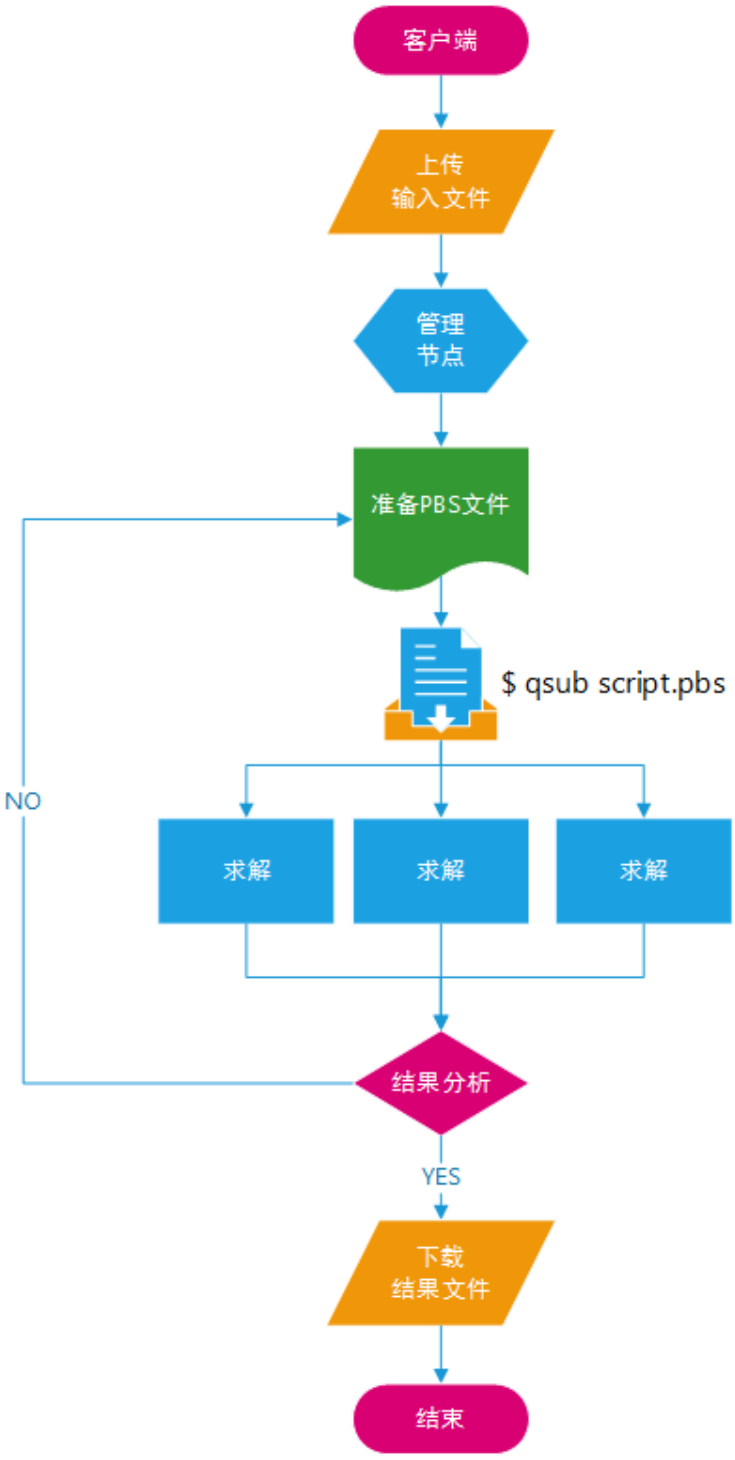


Fig. 1: HPC Workflow

用户运行程序的命令和 PBS 环境变量设置组成了 PBS 的作业脚本，基于 Intel MPI 的 vasp 范例脚本如下：

```
#!/bin/bash
#
#PBS -j oe
#PBS -q q1
#PBS -l select=1:ncpus=16
#PBS -l walltime=72:00:00
#PBS -N vasp
#PBS -V

module load vasp/5.4.4

cd $PBS_O_WORKDIR

mpirun vasp_std
```

提交作业：

```
qsub intelmpi.pbs
```

或指定所需要的节点数和 CPU 核数：

```
qsub -l select=1:ppn=20 -l walltime=72:00:00 intelmpi.pbs
```

提交作业后，可以通过 `qstat` 命令查看所交作业的状态及相关信息：

```
qstat
Job id            Name             User              Time Use S Queue
----------------  ---------------- ----------------  -------- - -----
18.mn01           STDIN            zhangtl           00:00:00 R q1
```

只查看自己的作业：

```
qstat -u test
```

删除作业：

```
qdel 18
```

6.4 PBS 基本命令

PBS 脚本里常用的控制命令：

Option	Description
#PBS -N myJob	Assigns a job name. The default is the name of PBS job script.
#PBS -l nodes=4:ppn=2	The number of nodes and processors per node.
#PBS -q queuename	Assigns the queue your job will use.
#PBS -l walltime=01:00:00	The maximum wall-clock time during which this job can run.
#PBS -o path/my.out	The path and file name for standard output.
#PBS -e path/my.err	The path and file name for standard error.
#PBS -j oe	merges the standard error with the standard output of the job.
#PBS -m b	Sends mail to the user when the job begins.
#PBS -m e	Sends mail to the user when the job ends.
#PBS -m a	Sends mail to the user when job aborts (with an error).
#PBS -r n	Indicates that a job should not rerun if it fails.
#PBS -V	Exports all environment variables to the job.

6.5 查询队列信息

在管理节点上执行如下命令，可查询当前系统的队列:

```
qstat -Qf
qstat -Qf q1
```

查询系统节点状态:

系统提供 `pestat` 命令查询各节点的作业情况，执行

```
$ pestat

 Total: 2680    Used:  16       Free:  2664
================================================================
Node       state      pmem         pcpu       tasks     freecpu
----------------------------------------------------------------
c001       free      128400         40         16          24
----------------------------------------------------------------
c002       free      128400         40          0          40
----------------------------------------------------------------
c003       free      128400         40          0          40
...
...
sg15       free       64232         12          0          12
----------------------------------------------------------------
sg16       free       64232         12          0          12
----------------------------------------------------------------
```

节点状态说明：* excl： 所有 CPU 资源已被占用; * busy： CPU 已接近满负荷运行; * free： 全部或部分 CPU 空闲; * offl： 管理员手动指定离线状态;

6.6 查询作业运行信息

```
qstat -an
```

查询作业命令 qstat 的参数可以为：

- -q：列出系统队列信息
- -B: 列出 PBS 服务器的相关信息
- -Q: 列出队列的一些信息
- -an: 列出队列中所有作业及分配的节点
- -r: 列出正在运行的队列
- -f jobid: 列出指定作业的详细信息
- -Qf queue: 列出指定队列的所有信息

6.7 删除作业

删除作业的命令：

```
$ qdel 123
```

123 为要删除的作业 id。

Note: note

1. 非 root 只能查看、删除自己的作业。
2. 在提交作业是，可以根据算例规模合理估算所需的时间和资源，有助于快速分配资源。
3. 如有紧急作业，请联系管理员处理系统当前正在运行的作业。

6.8 节点管理

节点状态可以用 pbsnodes 命令查看

```
[root@mn01 ~]# pbsnodes c001
c001
     Mom = c001
     Port = 15002
     pbs_version = 19.1.1
     ntype = PBS
     state = free
     pcpus = 40
     resources_available.arch = linux
     resources_available.host = c001
     resources_available.mem = 131482344kb
     resources_available.ncpus = 40
```

(continues on next page)

(continued from previous page)

```
     resources_available.vnode = c001
     resources_assigned.accelerator_memory = 0kb
     resources_assigned.hbmem = 0kb
     resources_assigned.mem = 0kb
     resources_assigned.naccelerators = 0
     resources_assigned.ncpus = 0
     resources_assigned.vmem = 0kb
     queue = q1
     resv_enable = True
     sharing = default_shared
     last_state_change_time = Sun Aug 18 16:36:08 2019
     last_used_time = Sun Sep  1 17:23:15 2019
```

可以通过 pbsnode -o nodename 标记节点为 OFFLINE，可以阻止节点接受作业。

可以通过 pbsnode -c nodename 标记节点为 ONLINE，让节点接受作业。

可以通过 pbsnode -a 列出所有节点

可以通过 pbsnode -l 列出所有被标记为 OFFLINE 的节点

6.9 队列权限管理

启用队列 ACL：

```
# qmgr -c "set queue workq acl_user_enable = True"
# qmgr -c "set queue workq acl_group_enable = True"
```

关闭：

```
# qmgr -c "set queue workq acl_user_enable = False"
# qmgr -c "set queue workq acl_group_enable = False"
```

添加队列访问权限:

```
#qmgr -c "set queue q1 acl_users = curly"
#qmgr -c "set queue q1 acl_users += jerry"
#qmgr -c "set queue q1 acl_users += larry"
```

删除队列访问权限

```
# qmgr -c "set queue q1 acl_users -= curly"
# qmgr -c "set queue q1 acl_users -= jerry"
```

6.10 节点与队列对应

默认的，节点属于所有队列，可以通过命令设置某节点归属于某个特定队列，命令如下：

```
# qmgr -c "set node c001 queue = q1"
```

To remove nodes from a specific queue

```
# qmgr -c "unset node node-name queue"
```

6.11 qmgr

The Qmgr Command

The PBS manager command qmgr provides a command-line administrator interface. The command reads directives from standard input. The syntax of each directive is checked and the appropriate request sent to the Server(s). A qmgr directive takes one of the following forms:

```
command server [names] [attr OP value[,...]]
command queue  [names] [attr OP value[,...]]
command node   [names] [attr OP value[,...]]
```

where command is the command to perform on an object

qmgr commands list below:

Command	Explanation
active	Set the active objects.
create	Create a new object, applies to queues and nodes.
delete	Destroy an existing object (queues or nodes).
set	Define or alter attribute values of the object.
unset	Clear the value of the attributes of the object.
list	List the current attributes and values of the object.
print	Print all the queue and server attributes.

CHAPTER 7

应用软件

7.1 vasp

7.1.1 vasp 版本

vasp 版本 **5.4.4**

编译器：Intel Fortran 2019.4

MPI：Intel MPI 2019.4

BLAS: Intel MKL 2019.4

7.1.2 vasp 路径

安装路径：`/share/apps/vasp/5.4.4/bin`

7.1.3 modulefile

加载 vasp module 变量模块，默认的，会加载所有依赖的环境变量，如 intel mpi。

```
module load vasp/5.4.4
```

7.1.4 vasp 脚本

```
#!/bin/bash
#
#PBS -j oe
#PBS -q q1
#PBS -l select=1:ncpus=24
#PBS -l walltime=72:00:00
#PBS -N vasp
#PBS -V

module load vasp/5.4.4

cd $PBS_O_WORKDIR

mpirun vasp_std 2>&1 | tee output.txt
```

7.1.5 提交 vasp 作业

用 qsub 提交：

```
$ qsub vasp.pbs
```

7.2 gromacs

7.2.1 gromacs 版本

gromacs 版本:

- gromacs 4.6.7
- gromacs 5.1.4

7.2.2 gromacs 路径

安装路径：`/share/apps/gromacs`

Version	precision	compiler & MPI
4.6.7-intel2019.4-double	double	intel 2019.4 + impi 2019.4
4.6.7-intel2019.4-single	single	intel 2019.4 + impi 2019.4
5.1.4-gcc910-double	double	gcc 9.1.0 + impi 2019.4
5.1.4-gcc910-single	single	gcc 9.1.0 + impi 2019.4

7.2.3 modulefile

加载 module 变量模块，默认的，会加载所有依赖的环境变量，如 intel mpi。

```
module avail gromacs

gromacs/4.6.7-intel2019.4-dp
gromacs/4.6.7-intel2019.4-sp
gromacs/5.1.4-gcc910-dp
gromacs/5.1.4-gcc910-sp
```

7.2.4 gromacs 脚本

```
#!/bin/bash
#PBS -q q1
#PBS -l nodes=1:ppn=16
#PBS -l walltime=12:00:00
#PBS -V
#PBS -j oe
#PBS -N gromacs

gromacs/4.6.7-intel2019.4-sp

# mdrun_d is the MPI version of mdrun.
export EXEC=mdrun_d
export WORKDIR=$PBS_O_WORKDIR
export NPROCS=`wc -l $PBS_NODEFILE |gawk '//{print $1}'`

cd $WORKDIR
mpirun -np $NPROCS `which $EXEC` -s fws_em.tpr \
       -o fws_em.trr -c fws_b4pr.pdb -e em.edr -g em.log
```

7.2.5 提交 gromacs 作业

用 qsub 提交：

```
$ qsub gromacs.pbs
```

7.3 Gaussian

7.3.1 G09 版本

Gaussian 版本 **G09RevC.01**

7.3.2 G09 路径

安装路径：/share/apps/gaussian/

要使用 G09，需要设置 g09root 变量：

```
export g09root=/share/apps/gaussian
source $g09root/g09/bsd/g09.profile
```

7.3.3 G09 脚本

```
#!/bin/bash
#
#PBS -q q1
#PBS -l select=1:ncpus=20
#PBS -l walltime=24:00:00
#PBS -N Gaussian
#PBS -j oe
#PBS -V

input=test397.com

cd $PBS_O_WORKDIR

# set env
export g09root=/share/apps/gaussian
source $g09root/g09/bsd/g09.profile

# prepare a local scratch dirs
scratch="/tmp/$USER.$PBS_JOBID"

if [ ! -d $scratch ]; then
    mkdir -p $scratch
fi
export GAUSS_SCRDIR=$scratch

# run Gaussian 09
time g09 $input

rm -rf $scratch
```

7.3.4 提交 gaussian 作业

用 qsub 提交：

```
$ qsub g09.pbs
```

7.4 Orca

7.4.1 orca 版本

orca 版本 **4.2.0**

MPI：OpenMPI 3.1.4

7.4.2 orca 路径

安装路径：/share/apps/orca/4.2.0

7.4.3 modulefile

加载 module 变量模块，默认的，会加载所有依赖的环境变量，如 OpenMPI。

```
module load orca/4.2.0
```

7.4.4 orca 测试算例 CASCF.inp

```
#
# Initial CASCF on [Cr(NH3)6]3+
#
# def2-TZVPP = triple zeta basis
# RI-JK = use the RI-JK approximation for the Fock matrix
# (not required - just used for more speed here)
# def2/JK = auxiliary basis for the RI approximation
# (automatically sets the auxiliary slots for
# the Fockian [AuxJK] and gradient integrals [AuxC])
# for Fock [AuxJK slot] and gradient integrals [AuxC slot])
#
# Conv = store integrals on disk (required for RIJK)
# XYZFile = leave coordinates on disk (convenient later)
# normalprint = slightly larger printing than default + includes Loewdin
# population analysis.
! def2-TZVPP def2/JK RI-JK Conv XYZFile PATOM
%pal
 nprocs 32
end
%casscf
nel 3 # number of active electrons
norb 5 # number of active orbitals
mult 4,2 # multiplicity blocks
nroots 10,9 # Roots per multiplicity blocks
end
* int 3 4
Cr      0       0       0       0       0       0
N       1       0       0       2.137   0       0
N       1       2       0       2.137   90      0
N       1       2       3       2.137   90      90
N       1       2       3       2.137   90      180
N       1       2       3       2.137   180     180
N       1       2       3       2.137   90      270
H       2       1       3       1.041   114     0
H       2       1       3       1.041   114     120
H       2       1       3       1.041   114     240
H       3       1       2       1.041   114     0
H       3       1       2       1.041   114     120
H       3       1       2       1.041   114     240
H       4       1       2       1.041   114     315
H       4       1       2       1.041   114     195
H       4       1       2       1.041   114     75
H       5       1       2       1.041   114     0
H       5       1       2       1.041   114     120
H       5       1       2       1.041   114     240
H       6       1       4       1.041   114     270
H       6       1       4       1.041   114     30
```

(continues on next page)

(continued from previous page)

```
H       6       1       4       1.041   114     150
H       7       1       2       1.041   114     45
H       7       1       2       1.041   114     165
H       7       1       2       1.041   114     285
*
```

7.4.5 orca 脚本

```
#!/bin/bash
#
#PBS -j oe
#PBS -q q1
#PBS -l select=1:ncpus=24
#PBS -l walltime=72:00:00
#PBS -N orca
#PBS -V

input=CASCF.inp
output=$(echo "${input%%.*}")
JobID=`echo $PBS_JOBID | cut -d"." -f1`

# Load Orca and openmpi modules
module load orca/4.2.0

cd $PBS_O_WORKDIR

export RSH_COMMAND="ssh -x"

# Define and create the scratch directry
SCRDIR="/tmp/$USER/$JobID"
mkdir -p "$SCRDIR"


# Get the full path to orca (helps with mpi problems)
ORCA_EXEC=$(which orca)

cp $input $SCRDIR

cd $SCRDIR
$ORCA_EXEC $input 2>&1 | tee $PBS_O_WORKDIR/${output}.out.$JobID

cp ${SCRDIR}/*.* $PBS_O_WORKDIR

rm -rf ${SCRDIR}
```

7.4.6 提交 orca 作业

用 qsub 提交：

```
$ qsub orca.pbs
```

查看作业输出：

```
...skip...

--------------------
Rotational spectrum
--------------------

Rotational constants in cm-1:     0.048707     0.048707     0.048707
Rotational constants in MHz :  1460.193311  1460.193311  1460.193311

 Dipole components along the rotational axes:
x,y,z [a.u.] :    -0.000000    -0.000930    -0.000930
x,y,z [Debye]:    -0.000000    -0.002364    -0.002364

Timings for individual modules:

Sum of individual times         ...      105.191 sec (=   1.753 min)
GTO integral calculation        ...        9.933 sec (=   0.166 min)   9.4 %
SCF iterations                  ...        2.505 sec (=   0.042 min)   2.4 %
CASSCF iterations               ...       92.752 sec (=   1.546 min)  88.2 %
                           ****ORCA TERMINATED NORMALLY****
TOTAL RUN TIME: 0 days 0 hours 1 minutes 46 seconds 180 msec
```

7.5 ABCluster

7.5.1 ABCluster 版本

ABCluster 版本 **4.2.0**

MPI：OpenMPI 3.1.4

7.5.2 ABCluster 路径

安装路径：/share/apps/ABCluster/4.2.0

7.5.3 modulefile

加载 module 变量模块，默认的，会加载所有依赖的环境变量，如 OpenMPI。

```
module load ABCluster/4.2.0
```

7.5.4 ABCluster 测试算例 cascf.inp

```
#
# Initial CASCF on [Cr(NH3)6]3+
#
# def2-TZVPP = triple zeta basis
# RI-JK = use the RI-JK approximation for the Fock matrix
```

(continues on next page)

(continued from previous page)

```
# (not required - just used for more speed here)
# def2/JK = auxiliary basis for the RI approximation
# (automatically sets the auxiliary slots for
# the Fockian [AuxJK] and gradient integrals [AuxC])
# for Fock [AuxJK slot] and gradient integrals [AuxC slot])
#
# Conv = store integrals on disk (required for RIJK)
# XYZFile = leave coordinates on disk (convenient later)
# normalprint = slightly larger printing than default + includes Loewdin
# population analysis.
! def2-TZVPP def2/JK RI-JK Conv XYZFile PATOM
%pal
 nprocs 32
end

%casscf
nel 3 # number of active electrons
norb 5 # number of active orbitals
mult 4,2 # multiplicity blocks
nroots 10,9 # Roots per multiplicity blocks
end

* int 3 4
Cr 0 0 0 0 0 0
N 1 0 0 2.137 0 0
N 1 2 0 2.137 90 0
N 1 2 3 2.137 90 90
N 1 2 3 2.137 90 180
N 1 2 3 2.137 180 180
N 1 2 3 2.137 90 270
H 2 1 3 1.041 114 0
H 2 1 3 1.041 114 120
H 2 1 3 1.041 114 240
H 3 1 2 1.041 114 0
H 3 1 2 1.041 114 120
H 3 1 2 1.041 114 240
H 4 1 2 1.041 114 315
H 4 1 2 1.041 114 195
H 4 1 2 1.041 114 75
H 5 1 2 1.041 114 0
H 5 1 2 1.041 114 120
H 5 1 2 1.041 114 240
H 6 1 4 1.041 114 270
H 6 1 4 1.041 114 30
H 6 1 4 1.041 114 150
H 7 1 2 1.041 114 45
H 7 1 2 1.041 114 165
H 7 1 2 1.041 114 285
*
```

7.5.5 ABCluster 脚本

```
#!/bin/bash
#
```

(continues on next page)

(continued from previous page)

```
#PBS -j oe
#PBS -q q1
#PBS -l select=1:ncpus=24
#PBS -l walltime=72:00:00
#PBS -N ABCluster
#PBS -V

input=IMF3.inp

# Load ABCluster and openmpi modules
module load ABCluster/4.2.0

cd $PBS_O_WORKDIR

export RSH_COMMAND="ssh -x"

# Get the full path to ABCluster (helps with mpi problems)
ABCluster_EXEC=$(which ABCluster)

$ABCluster_EXEC "$input" 2>&1 | tee| ABCluster.out
```

7.5.6 提交 ABCluster 作业

用 qsub 提交：

```
$ qsub ABCluster.pbs
```