中国国家网格 12 区

GPU 资源用户使用手册

2020 年 3 月

目录

1 系统资源简介 1

1.1 计算资源（节点配置） 1
1.2 存储资源（文件系统） 1

2 应用软件管理 2

2.1 使用 CNGRID12 已部署的应用软件及开发环境 2
2.1.1 简介 2
2.1.2 基本命令 2
2.1.3 已部署软件及开发环境列表 3
2.1.4 实例说明 5
2.2 软件安装/环境搭建 7
2.2.1 通过 Anaconda 进行自定义 python 环境安装 7
2.3 程序测试流程 9
2.4 提交作业 10
2.4.1 提交单卡计算任务 10
2.4.2 提交单机多卡计算任务 11
2.4.3 提交多机多卡计算任务 12
2.5 查看作业状态 13
2.6 取消作业 14

3 关键技术知识参考资料 14

3.1 SLURM 作业管理系统 14
3.1.1 基本概念 14
3.1.2 基本术语 14
3.1.3 三种作业提交模式区别 15
3.1.4 sinfo 查看系统资源 16
3.1.5 squeue 查看作业状态 18
3.1.6 srun 交互式提交作业 19
3.1.7 sbatch 后台提交作业 20
3.1.8 salloc 分配模式作业提交 21
3.1.9 scancel 取消已提交的作业 22
3.1.10 scontrol 查看正在运行的作业信息 22
3.1.11 sacct 查看历史作业信息 22

1 系统资源简介

1.1 计算资源（节点配置）

cngrid12 的 gpu 队列配置为：CPU 型号 Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz，单节点 20 核，内存大小为 128GB。通过 sinfo 查看队列情况，gpu 信息为 4 块 TESLA-V100 16G 显存。

1.2 存储资源（文件系统）

cngird12 文件系统被分为/home 、/dat01 两个分区，所有登录服务节点和计算节点都可以访问这三个分区，登陆系统后通过 pwd 命令可以查看自己当前所在的分区。

/home 分区：

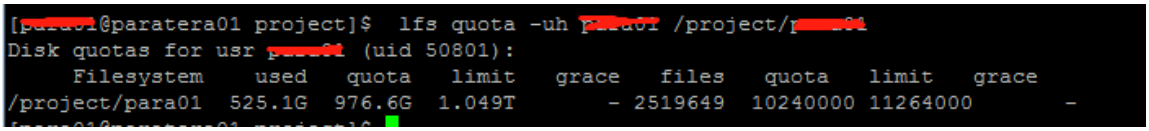
/home 分区的用户主目录为用户默认家目录，通常$HOME 的位置位于该分区下，家目录形式为“/home/超算账号”（如/home/para01）。该分区的磁盘容量较小仅用于存储用户的环境变量等信息，其磁盘限额为每用户 1GB。

/dat01 分区：

/dat01 分区为数据存放区域，主要用于项目文件和运行作业，该分区磁盘容量大，数据读写快。其磁盘限额为每用户 1TB。在脚本中使用“/dat01/系统账号”（如/dat01/para01）可引用该分区下的用户主目录，也可以使用“/home/超算账号/dat01”（如：/home/para01/dat01）访问该分区。

温馨提示：

1.可使用可用命令 lfs quota -uh 超算账号 /分区/超算账号 查看磁盘详细信息



2./home 和 /dat01 分区默认的存储是长期有效，建议您经常清理备份有效数据如果确实需要比较大的磁盘空间存储数据和执行程序，可以联系客户经理增加配额。

2 应用软件管理

该章节将对算例管理的全生命周期进行详细叙述，包括算例运行前的环境搭建、代码编译、算例提交、算例取消和查看算例状态。

2.1 使用 cngrid12 已部署的应用软件及开发环境

2.1.1 简介

“cngrid12”已部署多款应用软件及软件开发运行环境。

由于不同用户在“cngrid12”上可能需要使用不同的软件环境，配置不同的环境变量，软件之间可能会相互影响，因而在“cngrid12”上安装了 module 工具来对应用软件统一管理。module 工具主要用来帮助用户在使用软件前设置必要的环境变量。用户使用 module 加载相应版本的软件后，即可直接调用超算上已安装的软件。

2.1.2 基本命令

常用命令如下：

命令	功能	例子
module avail	查看可用的软件列表	
module load [modulesfile]	加载需要使用的软件	module load nvidia/cuda/10.0
module show [modulesfile]	查看对应软件的环境（安装路径、库路径等）	module show nvidia/cuda/10.0
module list	查看当前已加载的所有软件	
module unload [modulesfile]	移除使用 module 加载的软件环境	module unload nvidia/cuda/10.0

module 其它用法，可使用 module --help 中查询。module 加载的软件环境只在当前登陆窗口有效，退出登陆后软件环境就会失效。用户如果需要经常使用一个软件，可以把 load 命令放在~/.bashrc 或者提交脚本里面。

2.1.3 已部署软件及开发环境列表

已安装应用软件及 AI 框架包括包括：

序号	软件名称	版本
1	pytorch	1.0_python3.7_cpu
2	pytorch	1.0_python3.7_gpu
3	nwchem	6.8.1
4	openfoam	openfoam-6
5	lammps	201812/lmp_cuda_openmpi
6	lammps	201812/lmp_intelcpu_intelmpi
7	lammps	201812/lmp_intelknl_intelmpi
8	liggghts	3.8.0/lmp_gcc_openmpi
9	matlab	R2017b
10	matlab	R2018b(default)
11	matlab	R2019b
12	namd	2.1.3-cpu(default)
13	namd	2.1.3-gpu
14	netcdf	4.4.0
15	netcdf-fortran	4.4.0
16	tensorflow	python2.7-cpu
17	tensorflow	python2.7-gpu
18	tensorflow	python3.6-cpu
19	tensorflow	python3.6-gpu
20	gromacs	2018.3-cpu
21	gromacs	2019.1-knl

22	gromacs	2019.4-gpu
23	gromacs	4.6.7-cpu
24	hdf5	1.10.5
25	hdf5	1.8.16
26	hdf5	1.8.20
27	htslib	1.10.1
28	texlive	2019
29	bcftools	1.10.1
30	caffe	pycaffe
31	vtk	7.1.1_gcc7.3_openmpi3.1.3

已安装软件开发运行环境包括：

序号	软件名称	版本
1	anaconda	2.7
2	anaconda	3.7
3	R	3.5.1
4	R	3.6.1
5	dock	dock6.9
6	go	1.11.2
7	go	1.12
8	intelpython	2.7
9	intelpython	3.6
10	intel parallel studio	2013
11	intel parallel studio	2017u8
12	intel parallel studio	2019
13	java	jdk11
14	java	jdk8
15	cuda	10
16	cuda	10.1

17	cuda	9
18	cuda	9.1
19	cuda	9.2
20	openmpi	1.10.7
21	openmpi	2.1.5
22	openmpi	2.1.5_IB_gcc7.3
23	openmpi	2.1.5_OPA_gcc7.3_cuda9.0
24	openmpi	3.1.3_IB_gcc7.3_singularity3.0.1
25	openmpi	3.1.3_OPA_gcc7.3_cuda9.0_singularity3.0.1
26	openmpi	4.0.0_IB_gcc7.3_singularity3.0.1
27	openmpi	4.0.0_OPA_gcc6.3_cuda10.0_singularity3.1.0
28	openmpi	4.0.0_OPA_gcc7.3_cuda9.0_singularity3.0.1
29	perl	5.30.0
30	pgi	201810
31	pgi	201810openmpi
32	scl	gcc4.9
33	scl	gcc5.3
34	scl	gcc6.3
35	scl	gcc7.3
36	singularity	2.6.0
37	singularity	3.0.1
38	singularity	3.1.0
39	singularity	3.2.0

2.1.4 实例说明

（1）用户需要使用 cuda10.0 环境
操作步骤：

- 执行 module avail 查看系统中可用软件，查询到 cuda10.0 的 module 环境名称为 nvidia/cuda/10.0

```
$ module avail
bcftools/1.10.1                                  namd/2.1.3-gpu
caffe/pycaffe                                    netcdf/4.4.0
cran/R-3.5.1(default)                            netcdf-fortran/4.4.0
cran/R-3.6.1                                     nvidia/cuda/10.0
ctan/texlive/2019                                nvidia/cuda/10.1
dock/dock6.9                                     nvidia/cuda/9.0
google/go/1.11.2                                 nvidia/cuda/9.1
google/go/1.12                                   nvidia/cuda/9.2
google/tensorflow/python2.7-cpu                  nwchem/6.8.1
google/tensorflow/python2.7-gpu                  openfoam/openfoam-6
google/tensorflow/python3.6-cpu                  openmpi/1.10.7
google/tensorflow/python3.6-gpu                  openmpi/2.1.5
gromacs/2018.3-cpu
openmpi/2.1.5_IB_gcc7.3
gromacs/2019.1-knl
openmpi/2.1.5_OPA_gcc7.3_cuda9.0
gromacs/2019.4-gpu
```

- 执行 module load nvidia/cuda/10.0 加载 cuda10.0 环境
- 执行 module list 查看已加载的环境

```
$ module list
Currently Loaded Modulefiles:
  1) nvidia/cuda/10.0
```

（2）用户有 python 程序运行需要使用 python3.7 环境

- 执行 module avail 查看系统中可用软件，可以查询到系统中已预装 anaconda3.7（anaconda3.7 中已集成 python3.7.5），查询到 anaconda3.7 的 module 环境名称为 anaconda/3.7。
  注：Anaconda 是一个用于科学计算的 Python 发行版，支持 Linux, Mac, Windows 系统，提供了包管理与环境管理的功能，可以很方便地解决多版本 python 并存、切换以及各种第三方包安装问题。Anaconda 利用工具/命令 conda 来进行 package 和 environment 的管理，并且已经包含了 Python 和相关的配套工具。
- 执行 module load anaconda/3.7 加载 cuda10.0 环境
- 执行 module list 查看已加载的环境

```
$ module list
Currently Loaded Modulefiles:
  1) anaconda/3.7
```

（3）用户需要使用 pytorch 框架

执行 module avail 查看系统中可用软件，可以查询到系统中已预装 pytorch 框架，module 环境名称为 pytorch/1.0_python3.7_gpu

- 执行 module load pytorch/1.0_python3.7_gpu 加载 cuda10.0 环境
- 执行 module list 查看已加载的环境，可以发现 pytorch 运行所依赖的 cuda10.0 环境也被自动加载。

```
$ module list
Currently Loaded Modulefiles:
  1) nvidia/cuda/10.0                2) pytorch/1.0_python3.7_gpu
```

2.2 软件安装/环境搭建

如果执行 module avail 没有查询到所需要的软件，可以上传软件安装包，在 dat01（路径为:/home/超算账号/dat01）目录下自定义安装所需软件。

软件安装基本流程：

1．上传软件安装包。

2．打开【SSH】命令行窗口，cd 到对应目录下，进行软件、工具、库等安装部署和算例运行环境搭建。

2.2.1通过 Anaconda 进行自定义 python 环境安装

Anaconda 是一个用于科学计算的 Python 发行版，支持 Linux, Mac, Windows 系统，提供了包管理与环境管理的功能，可以很方便地解决多版本 python 并存、切换以及各种第三方包安装问题。Anaconda 利用工具/命令 conda 来进行 package 和 environment 的管理，并且已经包含了 Python 和相关的配套工具。

Conda 的环境管理功能允许我们同时安装若干不同版本的 Python，并能自由切换。假设我们加载 anaconda3.7 环境，那么 Python 3.7 就是默认的环境（默认名字是 base）。

假设我们需要安装 Python 3.4，此时，我们需要做的操作如下：

- 加载 anaconda3.7 环境

```
module load anaconda/3.7
```

- 创建一个名为 python34 的环境，指定 Python 版本是 3.4（不用管是 3.4.x，conda 会为我们自动寻找 3.4.x 中的最新版本）

注意：conda 环境会默认安装到 home 下的.conda 文件夹中，安装过程的安装包也会下载到此文件夹下，安装过程中会在~/.cache 目录下产生临时文件，

home 目录只有 1GB 空间，会导致空间不足。可在 dat01 下创建.conda 目录，在 home 下创建.conda 及.cache 目录的软连接来解决此问题。环境实际安装到了 dat01 目录下。

命令如下：

创建目录

```
mkdir ~/dat01/.conda
mkdir ~/dat01/.cache
```

创建软链接

```
ln -s   ~/dat01/.conda ~/.conda
ln -s   ~/dat01/.conda ~/.cache
```

创建名为 python34 的环境，指定 Python 版本是 3.4

```
conda create --name python34 python=3.4
```

- 查看已安装的环境

```
$ conda env list
输出如下,*代表目前激活的环境
# conda environments:
#
python34                  /home/yanjie/.conda/envs/python34
base                  *   /software/anaconda/anaconda3_5.3.1
```

- 安装好后，使用 activate 激活某个环境

```
source activate python34
```

激活后，会发现 terminal 输入的地方多了 python34 的字样，实际上，此时系统做的事情就是把默认 3.7 环境从 PATH 中去除，再把 3.4 对应的命令加入 PATH

- 此时，再次输入

```
python --version
```

可以得到`Python 3.4.5 :: Anaconda 4.1.1 (64-bit)`，即系统已经切换到了 3.4 的环境

- 如果想返回默认的 python 3.7 环境，运行：

```
source deactivate python34
```

Conda 的包管理：

- 安装 tensorflow-gpu 1.15

```
conda install tensorflow-gpu==1.15
或
pip install tensorflow-gpu==1.15
```

上述两个方法如果其中一个方法找不到对应包可以执行另一个方法安装，conda 或 pip 会从从远程搜索 tensorflow-gpu 1.15 的相关信息和依赖项目。

- 查看已经安装的 packages

```
conda list
```

- 最新版的 conda 是从 site-packages 文件夹中搜索已经安装的包，不依赖于 pip，因此可以显示出通过各种方式安装的包

注：如需客服协助安装、部署相关软件，可直接在支持服务群里面@在线客服。

2.3 程序测试流程

有多种方式可以将作业提交到计算节点进行计算。详见

本章节针对程序部署完成后的验证性测试、调试。推荐在不确定程序是否能够正常运行时使用此流程进行测试。测试程序成功运行后正式提交计算任务请参考 2.4 提交

1．使用 salloc 命令申请节点，命令格式 salloc -N 计算节点数量 -n 申请的核数 --gres=gpu:1 申请一块 GPU 卡 -p 分区名称。申请 gpu 分区一台服务器：

```
salloc -N 1 -n 5 --gres=gpu:1 -p gpu
输出如下:
$ salloc -N 1 -n 5 -p gpu --gres=gpu:1
salloc: Pending job allocation 7097279
salloc: job 7097279 queued and waiting for resources
salloc: job 7097279 has been allocated resources
salloc: Granted job allocation 7097279
salloc: Waiting for resource configuration
salloc: Nodes g0033 are ready for job
```

如上所示，输出提示 g0033 为我们申请的主机。

2．也可以执行 squeue 命令查看申请到的节点名称。

在当前终端执行：

```
squeue -u 用户名
```

或新开终端执行：

```
squeue
```

```
squeue -u user01
JOBID    PARTITION    NAME    USER    ST    TIME    NODES    NODELIST(REASON)
6720840        gpu    bash    user01    R    0:17        1        g0033
```

查看申请到的节点名称为 g0046

3．登录到 g0033 进行程序运行测试。

```
ssh g0033
```

4．程序运行过程中可使用 top 查看 CPU 利用率、使用 nvidia-smi 查看 GPU 利用率。

5．退出当前终端该作业停止或执行 scancel 作业 ID 停止该作业。

2.4 提交作业

本章节针对程序测试完成后的正式计算任务提交。推荐软件调试完毕后按照如下流程提交计算任务到计算节点进行计算。

编写提交脚本过程遇到困难时，可将程序测试能够成功运行的命令在群里发给客服，客服协助编写脚本。

注：1 颗 GPU 卡默认免费配置 5 个 CPU，2 块 GPU 配置 10 个 CPU，依次类推，超出该比例时，CPU 将会收取费用，单个 CPU 按照 1 块 GPU 合同价的五分之一计费。

2.4.1提交单卡计算任务

- 编辑作业提交脚本，文件名可自定义，此处命名为 submit-1GPU.sh，模版如下

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -n 5
#SBATCH -p gpu
#SBATCH --gres=gpu:1
#SBATCH --no-requeue
上面这部分为 sbatch 方式提交计算任务向 slurm 作业调度系统申请资源相关参数，此模版将申请 1 张 GPU 卡及 5CPU 核，单卡任务通常按此模版即可。如需其它参数可参考

此处可填写加载程序运行所需环境（根据软件需求，可使用 module load 、export 等方式加载）
```

```
此处可填写运行程序的命令
```

- 提交计算任务

在程序运行目录执行如下命令提交计算任务

```
sbatch submit-1GPU.sh
```

sbatch 命令没有屏幕输出，默认输出日志为提交目录下的 slurm-xxx.out 文件，可以使用 tail -f slurm-xxx.out 实时查看日志，其中 xxx 为作业号

2.4.2提交单机多卡计算任务

- 单机 2 卡任务。编辑作业提交脚本，文件名可自定义，此处命名为 submit-2GPU.sh，2 卡任务提交脚本模版如下

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -n 10
#SBATCH -p gpu
#SBATCH --gres=gpu:2
#SBATCH --no-requeue

上面这部分为 sbatch 方式提交计算任务向 slurm 作业调度系统申请资源相关参数，此模版将申请 2 张 GPU 卡及 10CPU 核，2 卡任务通常按此模版即可。如需其它参数可参考

此处可填写加载程序运行所需环境（根据软件需求，可使用 module load 、export 等方式加载）

此处可填写运行程序的命令
```

- 提交计算任务

在程序运行目录执行如下命令提交计算任务

```
sbatch submit-2GPU.sh
```

- 单机 3 卡任务。编辑作业提交脚本，文件名可自定义，此处命名为 submit-3GPU.sh，3 卡任务提交脚本模版如下

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -n 15
#SBATCH -p gpu
#SBATCH --gres=gpu:3
#SBATCH --no-requeue
```

```
上面这部分为 sbatch 方式提交计算任务向 slurm 作业调度系统申请资源相关参数，此模版将申请 3 张 GPU 卡及 15CPU 核，3 卡任务通常按此模版即可。如需其它参数可参考

此处可填写加载程序运行所需环境（根据软件需求，可使用 module load 、export 等方式加载）

此处可填写运行程序的命令
```

- 提交计算任务

在程序运行目录执行如下命令提交计算任务

```
sbatch submit-3GPU.sh
```

- 单机 4 卡任务。编辑作业提交脚本，文件名可自定义，此处命名为 submit-4GPU.sh，4 卡任务提交脚本模版如下

```
#!/bin/bash

#SBATCH -N 1
#SBATCH -n 20
#SBATCH -p gpu
#SBATCH --gres=gpu:4
#SBATCH --no-requeue

上面这部分为 sbatch 方式提交计算任务向 slurm 作业调度系统申请资源相关参数，此模版将申请 4 张 GPU 卡及 20CPU 核，单卡任务通常按此模版即可。如需其它参数可参考

此处可填写加载程序运行所需环境（根据软件需求，可使用 module load 、export 等方式加载）

此处可填写运行程序的命令
```

- 提交计算任务

在程序运行目录执行如下命令提交计算任务

```
sbatch submit-4GPU.sh
```

2.4.3提交多机多卡计算任务

以 2 机共 8 卡为例：

```
#!/bin/bash

#SBATCH -N 2
#SBATCH --ntasks-per-node=20
```

```
#SBATCH -p gpu
#SBATCH --gres=gpu:4
#SBATCH --no-requeue

上面这部分为 sbatch 方式提交计算任务向 slurm 作业调度系统申请资源相关参数，此模版将申请 2 台服务器 8 张 GPU 卡及 40CPU 核。如需其它参数可参考

此处可填写加载程序运行所需环境（根据软件需求，可使用 module load 、export 等方式加载）

此处可填写运行程序的命令,注意需和程序跨节点提交参数配合
```

2.5 查看作业状态

- 查看已提交的作业

```
$ squeue
JOBID    PARTITION NAME     USER     ST     TIME    NODES
NODELIST(REASON)
2011812              gpu          user01        R      4:39          1
g0011
```

其中，

第一列 JOBID 是作业号，作业号是唯一的。

第二列 PARTITION 是作业运行使用的队列名。

第三列 NAME 是作业名。

第四列 USER 是超算账号名。

第五列 ST 是作业状态，R（RUNNING）表示正常运行，PD（PENDING）表示在排队，CG（COMPLETING）表示正在退出，S 是管理员暂时挂起，CD（COMPLETED）已完成，F（FAILED）作业已失败。只有 R 状态会计费。

第六列 TIME 是作业运行时间。

第七列 NODES 是作业使用的节点数。

第八列 NODELIST(REASON)对于运行作业（R 状态）显示作业使用的节点列表；对于排队作业（PD 状态），显示排队的原因。

2.6 取消作业

执行 scancel 作业 ID 取消作业

```
$ scancel 2011812
```

3 关键技术知识参考资料

3.1 slurm 作业管理系统

Slurm（Simple Linux Utility for Resource Management，http://slurm.schedmd.com/）是开源的、具有容错性和高度可扩展大型和小型 Linux 集群资源管理和作业调度系统。超级计算系统可利用 Slurm 进行资源和作业管理，以避免相互干扰，提高运行效率。所有需运行的作业无论是用于程序调试还是业务计算均必须通过交互式并行 srun、批处理式 sbatch 或分配式 salloc 等命令提交，提交后可以利用相关命令查询作业状态等。请不要在登录节点直接运行作业（日常查看、编辑、编译等除外），以免影响其余用户的正常使用。

3.1.1 基本概念

Slurm 利用分区(partition)对 CPU、内存、网络等资源进行分类，以便将不同需求的作业运行到不同计算节点上。用户需利用 slurm 命令将该作业所需要的 CPU 核等资源等提交到特定的分区中，等作业申请的资源得到满足后，作业才开始运行。作业运行受分区、账户、服务质量(QOS)等限制。

3.1.2 基本术语

• user：用户名，一般为系统登录名，如 user01。

• account：账户，记账账户，多个用户可以在同一个账户下，一般为用户所在的组，如 task1。

• core：CPU 核，单颗 CPU 可以具有多颗 CPU 核。

• job：作业。

• job step：作业步，单个作业（job）可以有多个作业步。

• partition：分区(可理解为 LSF、PBS 等作业调度系统中的队列)。作业需在特定分区中运行，一般不同分区允许的资源不一样，比如单作业核数等。

• rank：秩，如 MPI 进程号。

• tasks：任务数，单个作业或作业步可有多个任务，一般一个任务需一个 CPU 核，可理解为所需的 CPU 核数。

• socket：CPU 插槽，可以简单理解为 CPU。

• stdin：标准输入文件，一般指可以通过屏幕输入或采用<文件名方式传递给程序的文件，对应 C 程序中的文件描述符 0。

• stdout：标准输出文件，程序运行正常时输出信息到的文件，一般指输出到屏幕的，并可采用>文件名定向到的文件，对应 C 程序中的文件描述符 1。

• stderr：标准出错文件，程序运行出错时输出信息到的文件，一般指也输出到屏幕，并可采用 2>定向到的文件（注意这里的 2），对应 C 程序中的文件描述符 2。

• <ENTITY>：实体

• <SPECS>：明细、规格

3.1.3 三种作业提交模式区别

• 批处理作业（采用 sbatch 命令提交）：

对于批处理作业，使用 sbatch 命令提交作业脚本，作业被调度运行后，在所分配的首个节点上执行作业脚本，在作业脚本中也可使用 srun 命令加载作业任务。

• 交互式作业提交（采用 srun 命令提交）：

资源分配与任务加载两步均通过 srun 命令进行：当在登录 shell 中执行 srun 命令时，srun 首先向系统提交作业请求并等待资源分配，然后在所分配的节点上加载作业任务。

• 分配模式作业（采用 salloc 命令提交）：

分配作业模式类似于交互式作业模式和批处理作业模式的融合。用户需要指定资源分配的需求条件，向资源管理器提出作业的资源分配请求。作业排队，当用户请求资源被满足时，将在用户提交作业的节点上执行用户所指定的命令，指定的命令执行结束后，也运行结束，用户申请的资源被释放。

salloc 后面如果没有跟定相应的脚本或可执行文件，则默认选择/bin/sh，用户获得了一个合适环境变量的 shell 环境。

salloc 和 sbatch 最主要的区别是 salloc 命令资源请求被满足时，直接在提交作业的节点执行相应任务。而 sbatch 则当资源请求被满足时，在分配的第一个节点上执行相应任务。

salloc 在分配资源后，再执行相应的任务，很适合需要指定运行节点和其它资源限制，并有特定命令的作业。

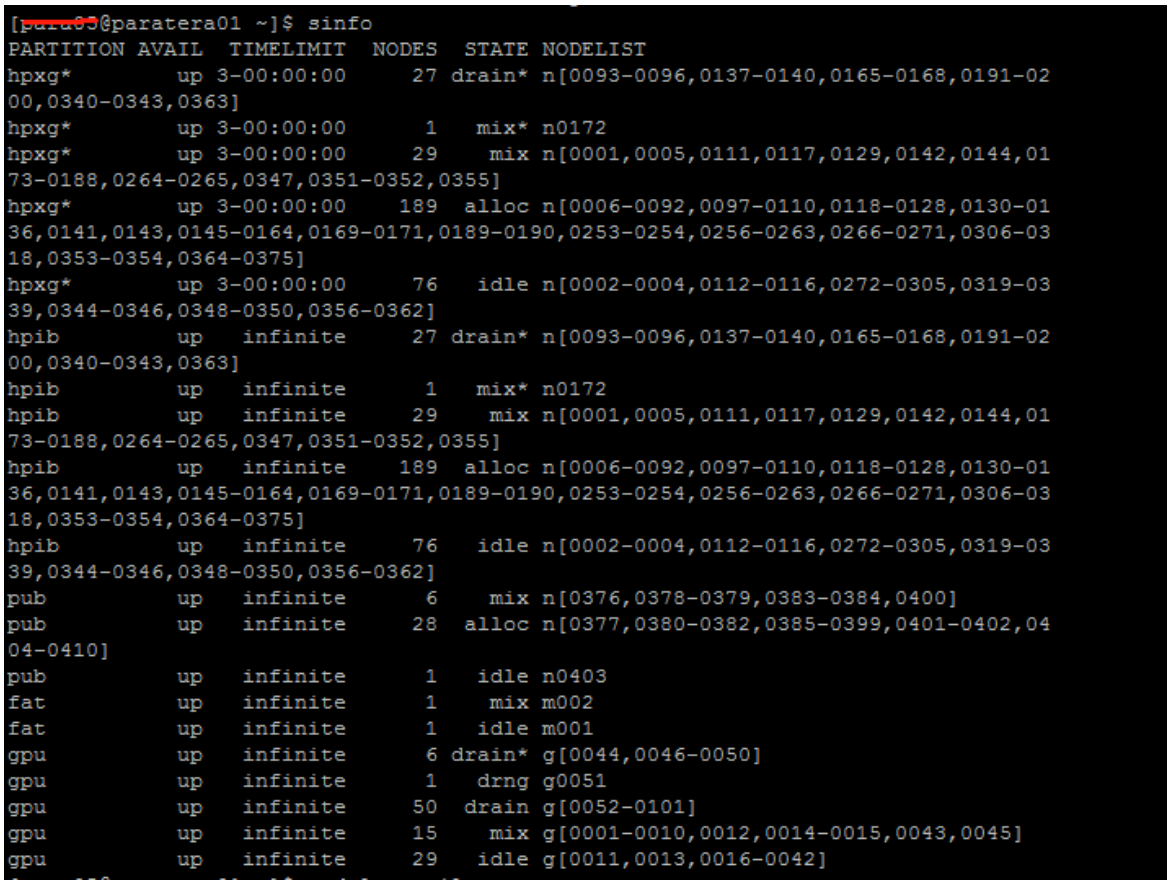
Cngird12 使用 Slurm 作业管理系统，采用节点共享模式。

作业管理系统常用命令如下：

命令	功能介绍	常用命令例子
sinfo	显示系统资源使用情况	sinfo
squeue	显示作业状态	squeue
srun	用于交互式作业提交	srun -N 2 -n 80 -p hpxg A.exe
sbatch	用于批处理作业提交	sbatch -N 2 -n 80 job.sh
salloc	用于分配模式作业提交	salloc -p gpu
scancel	用于取消已提交的作业	scancel JOBID
scontrol	用于查询节点信息或正在运行的作业信息	scontrol show job JOBID
sacct	用于查看历史作业信息	sacct -u para04 -S 03/01/17 -E 03/31/17 --field=jobid,partition,jobname,user,nnodes,start,end,elapsed,state

3.1.4 sinfo 查看系统资源

sinfo 得到的结果是当前账号可使用的队列资源信息，如下图所示：



```
[para03@paratera01 ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
hpxg*        up 3-00:00:00     27 drain* n[0093-0096,0137-0140,0165-0168,0191-02
00,0340-0343,0363]
hpxg*        up 3-00:00:00      1   mix* n0172
hpxg*        up 3-00:00:00     29    mix n[0001,0005,0111,0117,0129,0142,0144,01
73-0188,0264-0265,0347,0351-0352,0355]
hpxg*        up 3-00:00:00    189  alloc n[0006-0092,0097-0110,0118-0128,0130-01
36,0141,0143,0145-0164,0169-0171,0189-0190,0253-0254,0256-0263,0266-0271,0306-03
18,0353-0354,0364-0375]
hpxg*        up 3-00:00:00     76   idle n[0002-0004,0112-0116,0272-0305,0319-03
39,0344-0346,0348-0350,0356-0362]
hpib         up   infinite     27 drain* n[0093-0096,0137-0140,0165-0168,0191-02
00,0340-0343,0363]
hpib         up   infinite      1   mix* n0172
hpib         up   infinite     29    mix n[0001,0005,0111,0117,0129,0142,0144,01
73-0188,0264-0265,0347,0351-0352,0355]
hpib         up   infinite    189  alloc n[0006-0092,0097-0110,0118-0128,0130-01
36,0141,0143,0145-0164,0169-0171,0189-0190,0253-0254,0256-0263,0266-0271,0306-03
18,0353-0354,0364-0375]
hpib         up   infinite     76   idle n[0002-0004,0112-0116,0272-0305,0319-03
39,0344-0346,0348-0350,0356-0362]
pub          up   infinite      6    mix n[0376,0378-0379,0383-0384,0400]
pub          up   infinite     28  alloc n[0377,0380-0382,0385-0399,0401-0402,04
04-0410]
pub          up   infinite      1   idle n0403
fat          up   infinite      1    mix m002
fat          up   infinite      1   idle m001
gpu          up   infinite      6 drain* g[0044,0046-0050]
gpu          up   infinite      1   drng g0051
gpu          up   infinite     50  drain g[0052-0101]
gpu          up   infinite     15    mix g[0001-0010,0012,0014-0015,0043,0045]
gpu          up   infinite     29   idle g[0011,0013,0016-0042]
```

其中，

第一列 PARTITION 是队列名。

第二列 AVAIL 是队列可用情况，如果显示 up 则是可用状态；如果是 inact 则是不可用状态。

第三列 TIMELIMIT 是作业运行时间限制，默认是 infinite 没有限制。

第四列 NODES 是节点数。

第五列 STATE 是节点状态，idle 是空闲节点，alloc 是已被占用节点，comp 是正在释放资源的节点，其他状态的节点都不可用。

第六列 NODELIST 是节点列表。

sinfo 的常用命令选项：

命令示例	功能
sinfo -n g12345	指定显示节点 g12345 的使用情况
sinfo -p gpu	指定显示队列 gpu 情况

其他选项可以通过 sinfo --help 查询

3.1.5 squeue 查看作业状态

squeue 得到的结果是当前账号的作业运行状态，如果 squeue 没有作业信息，说明作业已退出。

```
$ squeue
$ JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
$ 2011812 gpu user01 R 4:39 1 g0011
```

其中，

第一列 JOBID 是作业号，作业号是唯一的。

第二列 PARTITION 是作业运行使用的队列名。

第三列 NAME 是作业名。

第四列 USER 是超算账号名。

第五列 ST 是作业状态，R（RUNNING）表示正常运行，PD（PENDING）表示在排队，CG（COMPLETING）表示正在退出，S 是管理员暂时挂起，CD（COMPLETED）已完成，F（FAILED）作业已失败。只有 R 状态会计费。

第六列 TIME 是作业运行时间。

第七列 NODES 是作业使用的节点数。

第八列 NODELIST(REASON)对于运行作业（R 状态）显示作业使用的节点列表；对于排队作业（PD 状态），显示排队的原因。

– AssociationJobLimit：作业达到其最大允许的作业数限制。
– AssociationResourceLimit：作业达到其最大允许的资源限制。
– AssociationTimeLimit：作业达到时间限制。
– BadConstraints：作业含有无法满足的约束。
– BeginTime：作业最早开始时间尚未达到。
– Cleaning：作业被重新排入分区，并且仍旧在执行之前运行的清理工作。
– Dependency：作业等待一个依赖的作业结束。
– FrontEndDown：没有前端节点可用于执行此作业。
– InactiveLimit：作业达到系统非激活限制。
– InvalidAccount：作业用户无效。
– InvalidQOS：作业QOS无效。
– JobHeldAdmin：作业被系统管理员挂起。
– JobHeldUser：作业被用户自己挂起。
– JobLaunchFailure：作业无法被启动，有可能因为文件系统故障、无效程序名等。

– Licenses：作业等待相应的授权。
– NodeDown：作业所需的节点宕机。
– NonZeroExitCode：作业停止时退出代码非零。
– PartitionDown：作业所需的分区出于DOWN状态。
– PartitionInactive：作业所需的分区处于Inactive状态。
– PartitionNodeLimit：作业所需的节点超过所用分区当前限制。
– PartitionTimeLimit：作业所需的分区达到时间限制。
– Priority：作业所需的分区存在高等级作业或预留。
– Prolog：作业的PrologSlurmctld前处理程序仍旧在运行。
– QOSJobLimit：作业的QOS达到其最大作业数限制。
– QOSResourceLimit：作业的QOS达到其最大资源限制。
– QOSTimeLimit：作业的QOS达到其时间限制。
– ReqNodeNotAvail：作业所需的节点无效，如节点宕机。
– Reservation：作业等待其预留的资源可用。
– Resources：作业等待其所需的资源可用。
– SystemFailure：Slurm系统失效，如文件系统、网络失效等。
– TimeLimit：作业超过去时间限制。
– QOSUsageThreshold：所需的QOS阈值被违反。
– WaitingForScheduling：等待被调度中。

squeue 的 常用命令选项：

命令示例	功能
squeue -j 123456	查看作业号为 123456 的作业信息
squeue -u para04	查看超算账号为 para04 的作业信息
squeue -p gpu	查看提交到 gpu 队列的作业信息
squeue -w c123	查看使用到 c123 节点的作业信息

其他选项可通过 squeue --help 命令查看

3.1.6 srun 交互式提交作业

srun [options] program 命令属于交互式提交作业，有屏幕输出，但容易受网络波动影响，断网或关闭窗口会导致作业中断。一般仅在调试程序时使用此方式提交作业。语法为：srun [OPTIONS...] executable [args...]

srun 命令示例：

```
srun -p hpxg -w g[1100-1101] -N 2 -n 80 -t 20 A.exe
```

交互式提交 A.exe 程序。如果不关心节点和时间限制，可简写为 srun -p gpu -n 80 A.exe

其中，

-p hpxg 指定提交作业到 hpxg 队列；

-w n[1100-1101] 指定使用节点 n[1100-1101]；

-N 2 指定使用 2 个节点；

-n 40 指定进程数为 40，

-t 20 指定作业运行时间限制为 20 分钟。

srun 的一些常用命令选项：

参数选项	功能
-N 3	指定节点数为 3
-n 12	指定进程数为 12，
-c 12	指定每个进程（任务）使用的CPU核为12，一般运行OpenMP等多线程程序时需，普通MPI程序不需要指定。
--gres=gpu:4	指定每台机器使用4张GPU卡。
-p gpu	指定提交作业到 gpu 队列
-w g[100-101]	指定提交作业到 g100、g101 节点
-x g[100,106]	排除 g100、g106 节点
-o out.log	指定标准输出到 out.log 文件
-e err.log	指定重定向错误输出到 err.log 文件
-J JOBNAME	指定作业名为 JOBNAME
-t 20	限制运行 20 分钟

srun 的其他选项可通过 srun --help 查看。

3.1.7 sbatch 后台提交作业

Slurm 支持利用 sbatch 命令采用批处理方式运行作业，sbatch 命令在脚本正确传递

给作业调度系统后立即退出，同时获取到一个作业号。作业等所需资源满足后开始运行。

sbatch 提交一个批处理作业脚本到 Slurm。批处理脚本名可以在命令行上通过传递给 sbatch，如没有指定文件名，则 sbatch 从标准输入中获取脚本内容。

脚本文件基本格式：

这种方式不受本地网络波动影响，提交作业后可以关闭本地电脑。sbatch 命令没有屏幕输出，默认输出日志为提交目录下的 slurm-xxx.out 文件，可以使用 tail -f slurm-xxx.out 实时查看日志，其中 xxx 为作业号。

sbatch 命令示例 1（40 个进程提交 A.exe 程序）：

编写脚本 job1.sh，内容如下：

```
#!/bin/bash
srun -n 40 A.exe
```

然后在命令行执行 sbatch -p gpu job1.sh 提交作业。脚本中的#!/bin/bash 是 bash 脚本的固定格式。从脚本的形式可以看出，提交脚本是一个 shell 脚本，因此常用的 shell 脚本语法都可以使用。作业开始运行后，在提交目录会生成一个 slurm-xxx.out 日志文件，其中 xxx 表示作业号。

sbatch 命令示例 2（指定 2 个节点，4 个进程，每个进程 12 个 cpu 核提交 A.exe 程序，限制运行 60 分钟）：

编写脚本 job2.sh，内容如下：

```
#!/bin/bash
#SBATCH -N 2
#SBATCH -n 4
#SBATCH -c 12
#SBATCH -t 60


A.exe
```

然后在命令行执行 sbatch -p gpu job2.sh 就可以提交作业。其中#SBATCH 注释行是 slurm 定义的作业执行方式说明，一些需要通过命令行指定的设置可以通过这些说明写在脚本里，避免了每次提交作业写很长的命令行。

3.1.8 salloc 分配模式作业提交

salloc 命令用于申请节点资源，一般用法如下：

```
1、执行 salloc -N 1 -p gpu，申请 1 台服务器资源；
2、执行 squeue 查看分配到的节点资源，比如分配到 g100；
```

3、执行 ssh g100 登陆到所分配的节点； 4、登陆节点后可以执行需要的提交命令或程序； 5、作业结束后，执行 scancel JOBID 释放分配模式作业的节点资源。

3.1.9 scancel 取消已提交的作业

scancel 可以取消正在运行或排队的作业。

scancel 的一些常用命令示例：

命令示例	功能
scancel 123456	取消作业号为 123456 的作业
scancel -n test	取消作业名为 test 的作业
scancel -p paratera	取消提交到 paratera 队列的作业
scancel -t PENDING	取消正在排队的作业
scancel -w c100	取消运行在 c100 节点上的作业

scancel 的其他参数选项，可通过 scancel --help 查看

3.1.10 scontrol 查看正在运行的作业信息

scontrol 命令可以查看正在运行的作业详情，比如提交目录、提交脚本、使用核数情况等，对已退出的作业无效。

scontrol 的常用示例：

```
scontrol show job 123456
```

查看作业号为 123456 的作业详情。

scontrol 的其他参数选项，可通过 scontrol --help 查看。

3.1.11 sacct 查看历史作业信息

sacct 命令可以查看历史作业的起止时间、结束状态、作业号、作业名、使用的节点数、节点列表、运行时间等。

sacct 的常用命令示例：

```
sacct -u para04 -S 2017-09-01 -E now --field=jobid,partition,jobname,user,nnodes,nodelist,start,end,elapsed,state
```

其中，-u para04 是指查看 para04 账号的历史作业，-S 是开始查询时间，-E 是截止查询时间，--format 定义了输出的格式，jobid 是指作业号，partition 是指提交队列，user 是指超算账号名，nnodes 是节点数，nodelist 是节点列表，start 是开始运行时间，end 是作业退出时间，elapsed 是运行时间，state 是作业结束状态。sacct --helpformat 可以查看支持的输出格式。

sacct 的其他参数选项可通过 sacct --help 查看。