宁夏超算云 N30 分区

GPU 资源用户使用手册

2022 年 8 月

目录

1 系统资源简介

1.1 计算资源（节点配置）

登录节点：主机名 ln01（32 核 256G），用于提供用户登录服务。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。

GPU 计算节点：主机命名规则：g0001-gxxxx，可参考 2.3 节内容申请 GPU 计算资源进行计算任务提交。

NC-N30 的 gpu 队列配置为：每台机器配置 8 块型号为 NVIDIA RTX3090 24GB 显存的 GPU，每块 GPU 卡默认分配 6 CPU 核（物理核）及 60GB 内存，即 GPU：CPU：内存配比为 1GPU 卡：6CPU 核：60GB 内存。不支持超核配置。如需更多 CPU 核数请提交到 gpu_c128 队列。

NC-N30 的 gpu_c128 队列配置为： 每台机器配置 8 块型号为 NVIDIA RTX3090 24GB 显存的 GPU，每块 GPU 卡默认分配 6 CPU 核（物理核）及 60GB 内存，即 GPU：CPU：内存配比为 1GPU 卡：6CPU 核：60GB 内存。支持超核配置，每卡最多可申请 16 CPU 核。CPU 核数超出每卡免费赠送的 6 个 cpu 核部分按 0.1 元/核时 计费。

登录节点可以联网，计算节点不能联网。如果程序运行过程需要下载数据请在登录节点下载好数据再提交作业到计算节点。如果有特殊需求请联系技术客服。

单作业允许申请 1~8 卡运行。gpu 分区不允许提交纯 CPU 作业。有特殊需求请与管理员联系开放 CPU 资源。

1.2 存储资源（文件系统）

NC-N30 根据数据的使用频次将数据分成三类：

家目录：用户登录上后即在家目录（home），用于存放用户环境设置等文件，不要在此目录下安装软件及存放文件，默认配额为 1GB。

作业运行数据存放目录：~/run 用于存放计算所需或访问频次较高的数据，读写性能较好，请将计算时需要使用的数据存储到该目录下并在此目录进行计算。

默认配额为 300GB。

温馨提示：

如果数据量超出磁盘配额，可以联系客户经理增加配额。

2 应用及软件管理

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 NC-N30 已部署的应用软件及开发环境

2.1.1 简介

“NC-N30”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“NC-N30”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“NC-N30”上安装了 module 工具用于统一管理应用软件。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	示例
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 cuda/11.1
module show [modulesfile]	查看对应软件的环境（安装路径、库路径等）	module show cuda/11.1
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 cuda/11.1

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录窗口有效，退出登录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

已安装软件开发运行环境包括但不限于下面所列：

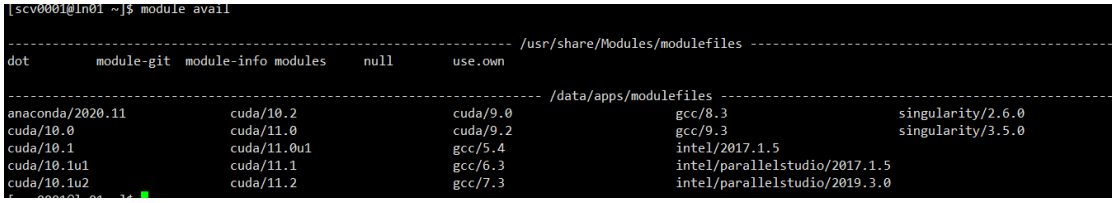
软件名称	版本
anaconda	2021.05
gcc	5.4、6.3、7.3、8.3、9.3
Intel Parallel Studio	2017.1.5、2019.3.0、2021
CUDA	11.1~11.4 全版本
其它软件	持续更新中

2.1.4 示例说明

用户需要使用 cuda11.1 环境

操作步骤：

- 执行 module avail 查看系统中可用软件，查询到 cuda11.1 的 module 环境名称为 cuda/11.1



- 执行 module load cuda/11.1 加载 cuda11.1 环境
- 执行 module list 查看已加载的环境

```
$ module list
Currently Loaded Modulefiles:
  1) cuda/11.1
```

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~/run 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】或者 putty 命令行窗口，cd 到对应目录下，进行软件、工具、库等安装部署和搭建算例运行所需环境。

2.2.1通过 Anaconda 进行自定义 python 环境安装

Anaconda 是一个开源的 Python 发行版本，其包含了 conda、Python 等 180 多个科学包及其依赖项。支持 Linux, Mac, Windows 系统，利用

conda 工具/命令来进行包管理与环境管理，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。并且已经包含了 Python 和相关的配套工具。

加载 anaconda/2020.11 环境后，默认的 python 是 3.8.5（环境名称为 base）。

假设我们需要安装 python3.7，此时，我们需要做的操作如下：

1) 加载 anaconda/2020.11 环境

```
module load anaconda/2020.11
```

2) 创建名为 py37 的环境，指定 Python 版本是 3.7

```
conda create --name py37 python=3.7
```

注：

a）不用管 python 的具体版本号，conda 会为我们自动寻找 3.7.x 中的最新版本

b）conda 环境会默认安装到~/.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。

c) 在自己创建的 conda 环境下使用 pip install 安装 python 包时请不要加上--user 参数，否则部署多个环境时会出现软件包冲突。

3) 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:

#

# conda environments:

#

base                  *  /data/apps/anaconda/2020.11

py37                     /data/home/username/.conda/envs/py37
```

4) 安装好后，使用 activate 激活某个环境

```
[username@ln01 ~]$ source activate py37

(py37) [username@ln01 ~]$ which  python

~/.conda/envs/py37/bin/python

(py37) [username@ln01 ~]$ python --version

Python 3.7.10
```

可以看到 Python 已切换到 3.7 的环境下。

5) 取消当前加载的环境，可执行：

```
$ source deactivate py37
```

6) 使用 conda 的包管理：

例如：安装最新版 pytorch，可执行

```
$ conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c
pytorch
或者：pip3 install torch torchvision torchaudio --extra-index-url
https://download.pytorch.org/whl/cu113
```

conda 会自动安装 pytorch 及其依赖组件。

7) 查看 conda 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，可直接在您专属的微信服务群里@客服。

2.3 提交计算任务

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。

使用本节方法提交计算任务后，调度系统会自动申请 GPU 资源，随后自动在计算节点执行用户所编辑的脚本内的命令，直到脚本执行结束作业自动退出（或者在作业运行时执行 scancel 命令取消作业后作业自动停止）。从申请到资

源至运行结束这段时间按卡时进行计费。每卡默认免费配置 6CPU 核及 60GB 内存。

gpu 队列计费公式：费用=卡数 x 时间（精确到秒）x 单价。

gpu_c128 队列计费公式：费用=卡数 x 时间（精确到秒）x 单价+（申请的 CPU 核数-6）x 时间（精确到秒）x 0.1 元/核时

每台机器配置 8 块型号为 NVIDIA RTX3090 24GB 显存的 GPU，每块 GPU 卡默认分配 6 CPU 核（物理核）及 60GB 内存，即 GPU：CPU：内存配比为 1GPU 卡：6CPU 核：60GB 内存。

2.3.1单节点任务提交

2.3.1.1 提交作业到 gpu 队列

gpu 队列每卡默认免费配置 6 核 CPU 及 60GB 内存，不支持超核配置。

作业提交命令：

sbatch --gpus=GPU 卡数 程序运行脚本

每个作业可以申请的 GPU 卡数范围为 1~8。不需要增加其它参数每卡默认分配 6 核及 60GB 内存。不加-p 参数默认提交到 gpu 队列，也可以添加-p gpu 指定提交到 gpu 队列。

申请到资源后程序必须按运行程序所需的参数指定进程数、线程数、卡数调用申请到的资源。注意：如果使用 srun 运行作业，必须使用-n 参数指定进

程数，或配合使用-c 参数指定每进程的线程数 （进程数 x 每进程线程数≤申请的核数）。

作业提交命令示例：

```
$ sbatch --gpus=1 ./run.sh
```

执行此命令后即申请到 1GPU 卡、6CPU 核、60GB 内存资源。作业显示为 R（Runing）状态（parajobs 命令查看作业状态）后即开始执行 run.sh 脚本中的内容。

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：

· 第一行以#!/bin/bash 等指定该脚本的解释程序，/bin/bash 可以变为 /bin/sh、/bin/csh 等。

· 在可执行命令之前的每行“#SBATCH”前缀后跟的参数作为作业调度系统参数。

默认，标准输出和标准出错都定向到同一个文件 *slurm-%j.out*，“%j”将被作业号 代替。

脚本 run.sh 示例 1，python 程序运行脚本示例：

sbatch --gpus=1 ./run.sh （申请 1 卡，6 核，60GB 内存）

```
#!/bin/bash
#SBATCH --gpus=1
#参数在脚本中可以加上前缀“#SBATCH”指定，和在命令参数中指定功能一致, 如果脚本中的参数和命令指定的参数冲突, 则命令中指定的参数优先级更高。在此处指定后可以直接 sbatch ./run.sh 提交。

#加载环境, 此处加载 anaconda 环境以及通过 anaconda 创建的名为 pytorch 的环境
module load anaconda/2021.05
source activate pytorch

#python 程序运行，需在.py 文件指定调用 GPU，并设置合适的线程数，batch_size 大小等
python train.py
```

脚本 run.sh 示例 2，HPC 程序运行脚本示例：

sbatch --gpus=2 ./run.sh　　(申请 2 卡，12 核，120GB 内存)

```
#!/bin/bash

#加载环境，此处加载编译程序使用的 intel parallelstudio 环境及 cuda 环境
```

```
module load intel/parallelstudio/2019.3.0
module load cuda/11.1


#程序运行，每卡分配 1 个进程 6 个线程
srun -n 2 -c 6  程序运行命令及参数
```

2.3.1.2 提交作业到 gpu_c128 队列

Gpu_c128 队列每卡默认免费配置 6 核 CPU 及 60GB 内存。支持超核配置，每卡最多可申请 16 CPU 核。CPU 核数超出每卡免费赠送 6 核部分按 0.1 元/核时 计费。

gpu_c128 队列主要提交参数解析：

-n <number> ，--ntasks=<number> ： 此作业申请<number>个进程数。如果不设置默认为 1。

-c <ncpus> ，--cpus-per-task=<ncpus> ： 每个进程需 ncpus 颗 CPU 核，一般运行 OpenMP 等多线程程序时需要设置。如果不设置默认为 1。

--gpus=GPU 卡数 ： 此作业申请的 GPU 卡数。

--gres=gpu:卡数 ： 每节点申请的 GPU 卡数，多机任务建议用此参数。

-p gpu_c128 :指定提交到 gpu_c128 队列，不指定默认提交到 gpu 队列。

注意：如果不设置-n、-c，每 GPU 卡会默认分配 6 核 CPU 供程序使用，但是默认的进程数和每进程分配的核数均为 1（仅影响 srun 运行程序的资源分配）。如果-n、-c 申请的核数少于 （6x 卡数） 也会分配 （6x 卡数） 个 CPU 核，大于 （6x 卡数） 则按实际申请为准。

作业提交命令示例 1：

申请 2GPU 卡，默认分配 12CPU 核及 120GB 内存。

```
$ sbatch --gpus=2 -p gpu_c128 ./run.sh
或
$ sbatch --gres=gpu:2 -p gpu_c128 ./run.sh
```

作业提交命令示例 2：

申请 2GPU 卡及 20CPU 核(srun 时申请 20 进程)，默认分配 120GB 内存。

```
$ sbatch --gpus=2 -n 20 -p gpu_c128 ./run.sh
或
$ sbatch --gres=gpu:2 -n 20 -p gpu_c128 ./run.sh
```

作业提交命令示例 3：

申请 2GPU 卡及 32CPU 核(srun 时申请 2 进程，每进程 16 核)，默认分配 120GB 内存。

```
$ sbatch --gpus=2 -n 2 -c 16 -p gpu_c128 ./run.sh
或
$ sbatch --gres=gpu:2 -n 2 -c 16 -p gpu_c128 ./run.sh
```

以上参数均可以在脚本中可执行命令之前的每行“#SBATCH”前缀后跟的参数作为作业调度系统参数。

2.3.2 跨节点任务提交

跨节点主要提交参数解析：

-N, –nodes=<node>：采用特定节点数运行作业，注意，这里是节点数，不是CPU 核数，实际分配的是节点数×每节点 CPU 核数。

--ntasks-per-node=<ntasks>：每个节点运行<ntasks>个进程，需与-N 配合使用。

-n <number> ，--ntasks=<number> ： 此作业申请<number>个进程数。如果不设置默认为 1。

-c <ncpus> ，--cpus-per-task=<ncpus> ： 每个进程需 ncpus 颗 CPU 核，一般运行 OpenMP 等多线程程序时需要设置。如果不设置默认为 1。

--gres=gpu:卡数 ： 每节点申请的 GPU 卡数，多机任务建议用此参数。

--qos=gpugpu ： 跨节点任务必须指定此参数，否则没有跨节点提交权限。

提交命令示例 1：

```
sbatch -N 2 --gres=gpu:8 --qos=gpugpu 脚本名
```

-N 2 申请 2 台机器

--gres=gpu:8 每台机器申请 8 卡。--gres=gpu:卡数代表每台机器申请多少卡，--gpus=卡数 代表总共申请多少卡，并且在多机场景--gpus=申请的卡数是随机分布的，不建议使用此参数。

--qos=gpugpu 申请开通跨节点权限后可以添加此参数进行跨节点任务提交。

CPU 默认每卡分配 6 核。

提交命令示例 2：

```
sbatch -N 2 --gres=gpu:8 --ntasks-per-node=48 --qos=gpugpu 脚本名
```

每节点申请 48 核（srun 时申请 48 进程）

提交命令示例 3：

```
sbatch -N 2 --gres=gpu:8 --ntasks-per-node=8 -c 6 --qos=gpugpu 脚本名
```

每节点申请 48 核（srun 时申请 8 进程,每进程 6 核）

30 区使用 NCCL 跨节点通信须添加下面参数：

```
export NCCL_DEBUG=INFO
export NCCL_IB_DISABLE=0
export NCCL_IB_HCA=mlx5_bond_0
export NCCL_SOCKET_IFNAME=bond0
export NCCL_IB_GID_INDEX=3
```

30 区 openmpi 跨节点：

不需要添加额外参数。

30 区 Intelmpi 跨节点：

不需要添加额外参数。

2.3.3 包节点用户提交作业到专属 vip 队列

申请包节点后会根据用户包的节点数量创建一个专属队列，队列命名规则：

vip_gpu_用户名,例如：vip_gpu_scz0002

提交任务时必须添加下面参数指定提交到专属 vip 队列:

```
-p vip_gpu_用户名
```

单机任务提交举例（提交一个 8 卡任务，脚本名为 run.sh）：

```
sbatch --gpus=8 -p vip_gpu_用户名  ./run.sh
```

跨节点(多机)任务提交参考 2.3.2 ,注意添加参数：-p vip_gpu_用户名

注意：为了防止误提交到公共队列产生费用，包节点用户会将公共队列提交权限关闭。请注意提交到 vip_gpu_用户名 队列才能正常运行。有额外的资源需求也可申请开通公共队列。

2.3.4高级功能：将数据集放到内存文件系统进行计算

30 区提供内存文件系统供用户使用，用户可以将数据集放到内存中直接读取，减少数据读取时间，最大程度降低 IO 瓶颈，发挥出 GPU 最大性能。内存文件系统使用的是每台机器（节点）上的内存，最大容量为 400GB，同时受用户提交任务申请的内存大小限制，例如：申请了 1 卡，最多能用 60GB，申请 8 卡，最多能用 400GB。

操作步骤：

（1）打包数据集

打包后解压到内存会比直接拷贝快很多,cd 到数据集目录所在路径，假如 datasets 是数据集的目录，则输入如下命令,也可以是其它名称。-cf 是打包不压缩，不建议加-z 参数压缩，压缩后解压时间会增加很多。

```
tar -cf datasets.tar datasets
```

注意 cd 到数据集的目录下打包，打包文件路径不建议写绝对路径，绝对路径打包解压后也会加上绝对路径。

（2）在提交脚本中运行程序之前加上下面代码，此代码是解压数据集到内存中

```
date
tar -xf datasets.tar -C /dev/shm
date
```

要注意解压后的文件夹路径，按照上述打包方式解压出的路径为 /dev/shm/datasets/

（3）将 python 代码中的读取数据集的路径改为/dev/shm/datasets （如果目录是其它名称按实际情况来写，注意要写解压到内存后的路径/dev/shm)

（4）sbatch 提交任务

如果有疑问可联系技术支持工程师协助。

2.4 查看作业状态

- 查看已提交的作业

```
$ parajobs
JOBID   PARTITION   NAME   USER ST   TIME   NODES NODELIST(REASON)
 9987110          gpu .jobscri paraai_t   R          1:52         1 g0028
     9987110 作业 GPU 利用率为:
g0028: pci.bus_id, utilization.gpu [%], utilization.memory [%],
memory.total [MiB], memory.free [MiB], memory.used [MiB]
g0028: 00000000:85:00.0, 0 %, 0 %, 16160 MiB, 16160 MiB, 0 MiB
```

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。

2.5 取消作业

执行 scancel 作业 ID 取消作业

```
$ scancel 2011812
```

2.6 程序测试流程（正式提交任务请参考 2.3）

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。此方法终端中断后程序作业即退出运行，正式提交作业请不要使用此方法。测试程序成功运行后正式提交计算任务请参考 2.3 提交计算任务

此方法提交有如下弊端：

1、salloc 方式申请资源后资源使用时长和任务实际运行时长没有关联，容易出现由于网络不稳定等问题导致的中断，或计算完毕后忘记取消作业等问题导致的费用浪费。建议一般不要使用此方式！

2、如果用 salloc 方法申请资源运行程序，程序运行完成后作业不会自动停止，直到退出终端或 scancel 取消作业后作业才会停止，才会停止计费。sbatch 命令提交，任务运行完成后作业立刻停止。

3、终端断开可能会导致运行的任务中断，如果作业未运行完成时终端断开会导致任务运行中断。sbatch 命令提交作业终端断开连接作业依然在后台运行直到结束。

4、如果多个作业提交到相同机器，登录机器上只能看到最后一个提交作业申请的卡。

使用下面方法可以申请 GPU 资源，然后直接登录到计算节点上进行计算。成功申请到资源后即开始计费。

1. 使用 salloc 命令申请 GPU。

```
$ salloc --gpus=1
salloc: Granted job allocation 313
salloc: Waiting for resource configuration
salloc: Nodes g0001 are ready for job
```

如上所示，输出提示 g0001 为我们申请的主机，之后可以 ssh g0001 登录到节点进行计算。

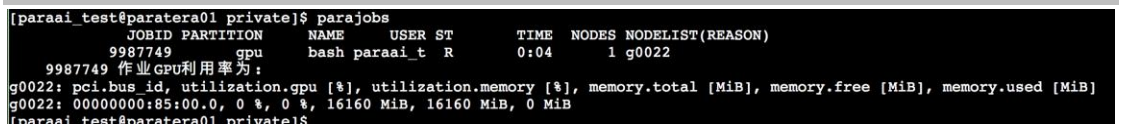
2. 执行 scancel 作业 ID 取消该作业（释放资源并停止计费）

```
$ scancel 313
```

3. 也可以执行 parajobs 命令查看申请到的节点名称。

执行：

```
parajobs
```




```
$ parajobs
JOBID   PARTITION    NAME     USER ST    TIME   NODES NODELIST(REASON)
9987749          gpu       bash    paraai_t     R          0:04         1   g0022
     9987749  作业 GPU 利用率为:
g0022: pci.bus_id, utilization.gpu [%], utilization.memory [%], memory.total [MiB],
memory.free [MiB], memory.used [MiB]
g0022: 00000000:85:00.0, 0 %, 0 %, 16160 MiB, 16160 MiB, 0 MiB
```

查看申请到的节点名称为 g0022，及 GPU 利用率状态。

4. 登录到 g0022 进行程序运行测试。

```
ssh g0022
```

5. 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。

6. 执行 scancel 作业 ID 停止该作业。