上海科技大学
ShanghaiTech University

高性能集群的介绍与使用
-slurm调度系统

上海科技大学图书信息中心

2023年4月

SLURM作业调度系统

- 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。

使用规范

1. 登录节点用于任务提交，环境配置（编译）等；禁止在登录节点运行计算程序。

2. 集群通过slurm调度系统实现作业管理、任务分配、调度等功能，所有计算任务应由slurm提交至计算节点。

3. 如使用salloc命令占用节点资源，请确保资源分配后会及时运行作业，避免长时间空卡。

slurm基本命令

Slurm命令	功能
sinfo	查看集群分区状态
squeue	查看作业
srun, salloc	交互式运行作业
sbatch	提交作业
scancel	取消作业
sacct	查看已完成作业

```
sinfo #查看所有分区状态
sinfo -l #查看更多信息
sinfo -N #查看节点状态
sinfo -N --states=idle #查看空闲节点
sinfo --partition=队列名 #查看队列名
sinfo –p 队列名，队列名
sinfo -n node-name #查看指定节点状态，多节点用逗号隔开
sinfo –i <seconds>  # 每隔相应的秒数，对输出的分区节点信息进行刷新

sinfo --help #查看sinfo的说明
```

alloc	idle	mix	down	drain
节点在用	节点可用	部分占用	节点下线	节点故障

查看作业

```
squeue #查看运行中的作业列表
squeue -l #查看作业详细信息
squeue -j job-id #查看特定作业信息
squeue --state=R #查看特定状态的作业信息
squeue –help #查看squeue的说明
```

作业状态

R	PD	CG	CD
正在运行	正在排队	即将完成	已完成

[Slurm Workload Manager - Documentation (schedmd.com)]
Slurm(Simple Linux Utility for Resource Management)
http://slurm.schedmd.com/

三种交作业的模式

– sbatch：批处理提交，提交后无需等待立即返回命令行终端。

– salloc：为需实时处理的作业分配资源，提交后等获得作业分配的资源后运行，作业结束后返回命令行终端。

– srun：运行并行作业，等获得作业分配的资源并运行，作业结束后返回命令行终端，实时交互式运行并行作业，一般用于测试，或者与sallcoc及sbatch结合

快速入门

- 两种运行作业的方法：

1. 将计算过程写成脚本，通过sbatch指令提交到计算节点执行；

2. 通过salloc申请到计算节点，再ssh连接到计算节点进行计算；

- 通过sinfo查看计算节点空闲状态；
- 可以通过squeue查看已经提交作业的排队情况；
- 通过scontrol show job 和sacct查询作业的相关信息；
- 通过scancel取消已经提交的作业；

sbatch作业提交

将整个计算过程，写到脚本中，通过sbatch指令提交到计算节点上执行；

1.脚本名称:job.sh

2.脚本内容：

```
#!/bin/bash
#SBATCH -o job.%j.out    # 脚本执行的输出将被保存在job.%j.out文件下，%j表示作业号;
#SBATCH –p bme_quick    # 作业提交的指定分区为bme_quick;
#SBATCH -J myJob          # 作业在调度系统中的作业名为myJob;
#SBATCH --nodes=1         # 申请节点数为1,如果作业不能跨节点(MPI)运行, 申请的节点数应不超过1;
#SBATCH --ntasks-per-node=1 # 每个节点上运行一个任务，默认一情况下也可理解为每个节点使用一个核心，如果程序不支持多线程(如openmp)，这个数不应该超过1；
hostname   #运行命令
```

3.提交命令：sbatch job.sh

sbatch作业提交

脚本常用参数：

-D, --chdir=<directory>　　# 指定工作目录；
--get-user-env　# 获取当前的环境变量；
--gres=<list>　# 使用gpu这类资源，如申请两块gpu则--gres=gpu:2
-J, --job-name=<jobname>　# 指定该作业的作业名；
-n=<number>　# 总进程数
-N,--nodes=<numberi>　#总节点数
--ntasks-per-node=<number>　# 每个节点的进程/任务数
-o, --output=<filename pattern>　# 输出文件，作业脚本中的输出如未指定将会输出到该文件；
-p, --partition=<partition_names>　# 将作业提交到对应分区；
-t, --time=<time>　# 允许作业运行的最大时间
-w, --nodelist=<node name list>　# 指定申请的节点；
-x, --exclude=<node name list>　# 排除指定的节点；
--help　　# 显示帮助信息；

sbatch脚本编写

跨节点多核心

```
#!/bin/bash
#SBATCH --job-name=jobname
#SBATCH -D ./
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=32
#SBATCH -o output.%j
#SBATCH -e error.%j
#SBATCH --time=1:00:00  /3:00
#SBATCH --partition=dm_debug
#SBATCH --gres=gpu:4   # 如果使用gpu的话
module load compiler/intel/composer_xe_2015.2.164  #导入MPI运行环境
module load mpi/intelmpi/5.0.2.044  #导入MPI运行环境
export VASP=/hpc/data/home/hpc_train/vasp/vasp.5.4.4/bin/vasp_std  #指定MPI应用程序
ulimit -s unlimited  #防止堆栈溢出
srun hostname -s | sort -n > myhosts  # 生成 machinefile
date >> info.dat
mpirun -n 64 -machinefile myhosts $VASP >> info.dat  # 执行MPI并行计算程序
date >> info.dat
```

sbatch脚本编写

GPU作业例子:

```
#!/bin/bash
#SBATCH -o job.%j.out
#SBATCH -e job.%j.err
#SBATCH -J myGPUJob
#SBATCH --partition=bme_gpu
#SBATCH --nodes=1     / #SBATCH -N=1
#SBATCH --ntasks-per-node=6
#SBATCH --gres=gpu:1     #每个节点上申请一块GPU
nvidia-smi
#SBATCH -N 1 -n 6 -p bme_gpu --gres=gpu:a100:2
```

脚本实例-pytorch

1.安装conda

2.创建虚拟环境并安装pytorch（1.6.0）

conda create -n pytorch-1.6.0 pytorch=1.6.0 torchvision cudatoolkit=10.1 -c pytorch

3.进入和推出创建好的虚拟环境

source activate pytorch-1.6.0

conda deactivate

4.提交作业

1）创建工作目录并进入

2）上传运行pytorch的相关文件/自己编辑一个test.py

```
import torch
print(torch.cuda.is_available())
torch.zeros(1).cuda()
```

3) 编写作业脚本：pytorchjob.sh

4) 提交作业：sbatch pytorchjob.sh

作业脚本：pytorchjob.sh

```
#!/bin/bash
#SBATCH -o job.%j.out
#SBATCH --partition=bme_gpu
#SBATCH -J pytorchjob
#SBATCH -N 1
#SBATCH --ntasks-per-node=2
#SBATCH --gres=gpu:1

source activate pytorch-1.6.0
python test.py
```

脚本实例-tensorflow

1.安装conda

2.创建虚拟环境并安装tensorflow GPU版本（1.6.0）

conda create -n tensorflow-gpu-2.6.0 tensorflow-gpu==2.6.0 -y

3.进入和推出创建好的虚拟环境

source activate tensorflow-gpu-2.6.0

conda deactivate

4.提交作业

1）创建工作目录并进入

2）上传运行tesndorflow的相关文件/自己编辑一个test.py

```
import tensorflow as tf
gpu_device_name = tf.test.gpu_device_name()
print(gpu_device_name)
```

3) 编写作业脚本：tensorflowjob.sh

4) 提交作业：sbatch tensorflowjob.sh

作业脚本：tensorflowjob.sh

```
#!/bin/bash
#SBATCH -o job.%j.out
#SBATCH --partition=bme_gpu
#SBATCH -J tensorflowjob
#SBATCH -N 1
#SBATCH --ntasks-per-node=2
#SBATCH --gres=gpu:1

source activate tensorflow-gpu-2.6.0
python test.py
```

salloc交互运行作业

申请计算节点，然后登录到申请到的计算节点上运行指令；

```
salloc -p bme_quick -N1 -n6 -t 2:00:00
```

salloc 申请成功后会返回申请到的节点和作业ID等信息，例如：申请到bme_comput17节点，作业ID为490891

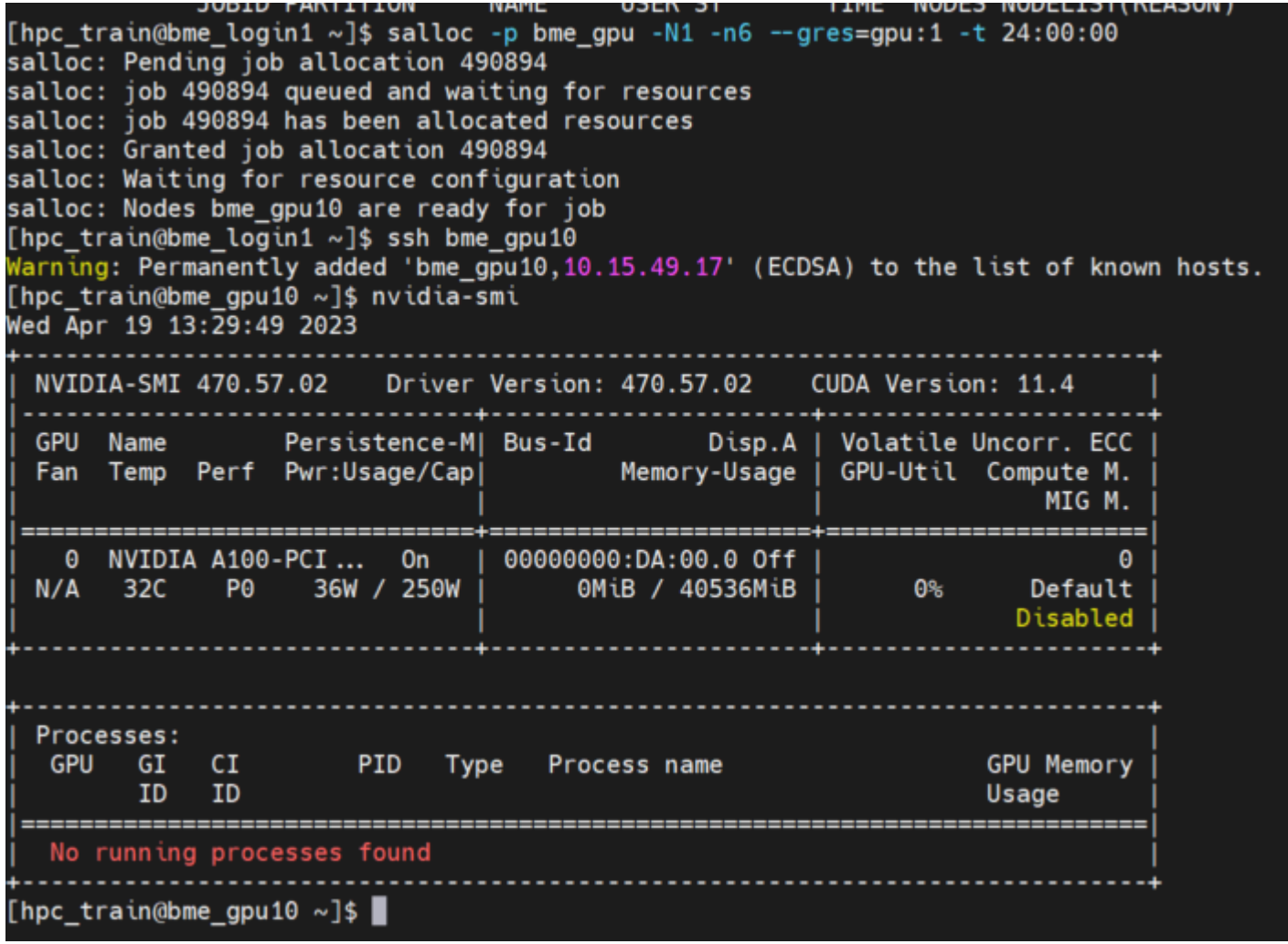
```
ssh bme_comput17      # 直接登录到刚刚申请到的节点bme_comput17调式作业
scancel 490891        # 计算资源使用完后取消作业
squeue -j 490891      # 查看作业是否还在运行，确保作业已经退出，避免浪费资源
```

```
[hpc_train@bme_login1 ~]$ salloc -p bme_quick -N1 -n6 -t 2:00:00
salloc: Pending job allocation 490891
salloc: job 490891 queued and waiting for resources
salloc: job 490891 has been allocated resources
salloc: Granted job allocation 490891
salloc: Waiting for resource configuration
salloc: Nodes bme_comput17 are ready for job
[hpc_train@bme_login1 ~]$ ssh bme_comput17
Warning: Permanently added 'bme_comput17,10.15.49.58' (ECDSA) to the list of known hosts.
[hpc_train@bme_comput17 ~]$ hostname
bme_comput17
[hpc_train@bme_comput17 ~]$ scancel 490891
[hpc_train@bme_comput17 ~]$ salloc: Job allocation 490891 has been revoked.
                                                                Killed by signal 1.
[hpc_train@bme_login1 ~]$ squeue -j 490891
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
[hpc_train@bme_login1 ~]$
```

salloc交互运行作业

GPU节点

```
salloc -p bme_gpu -N1 -n6 --gres=gpu:1 -t 3:00
# 例如申请成功后返回的作业号为490894，申请到的节点是bme_gpu10
ssh bme_gpu10
# 登录到bme_gpu10上调式作业
scancel 490894
# 计算结束后结束任务
squeue -j 490894
# 确保作业已经退出
```



salloc交互运行作业

跨节点

```
salloc -p bme_quick -N2 --ntasks-per-node=12 -t 2:00:00
# salloc 申请成功后会返回申请到的节点和作业ID等信息，假设申请到的是bme_gpu[06-07]节点，作业ID为531987
# 这里申请两个节点，每个节点12个进程，每个进程一个核心
# 根据需求导入MPI环境
module load mpi/intelmpi/5.0.2.044
# 根据以下命令生成MPI需要的machine file
srun hostname -s | sort -n > slurm.hosts
mpirun -np 24 -machinefile slurm.hosts hostname
# 结束后退出或者结束任务
scancel 531987
# 确保作业已经退出
squeue -j 531987
```

查看作业信息

1.squeue：显示队列中自己的作业

```
[hpc_train@bme_login1 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```

作业号，分区，作业名，用户，作业状态，运行时间，节点数量，运行节点(如果还在排队则显示排队原因

2.scontrol show job：显示正在运行或者刚结束没多久的作业信息

3.sacct：查询已结束作业的相关信息

```
[hpc_train@bme_login1 ~]$ sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
490891             bash  bme_quick  hpc_train          6 CANCELLED+      0:0
490891.exte+     extern             hpc_train          6  COMPLETED      0:0
490894             bash    bme_gpu  hpc_train          6 CANCELLED+      0:0
490894.exte+     extern             hpc_train          6  COMPLETED      0:0
[hpc_train@bme_login1 ~]$
```

作业号，作业名，分区，账户，申请的CPU数量，状态，结束代码

4.scancel：取消作业

scancel jobid

scancel -n jobname　scancel –p 分区名　scancel –t PENDING/RUNNING　scancel --help

如何申请合适的资源

显示队列、节点信息：sinfo -l

```
Thu Oct 20 13:01:57 2022
PARTITION      AVAIL  TIMELIMIT   JOB_SIZE ROOT OVERSUBS     GROUPS  NODES       STATE NODELIST
bme_gpu           up 5-00:00:00 1-infinite   no       NO        all     15       mixed bme_gpu[01-09,11-12,14-17]
bme_gpu           up 5-00:00:00 1-infinite   no       NO        all      2   allocated bme_gpu[10,13]
bmegpu_v100       up 5-00:00:00 1-infinite   no       NO        all      3       mixed bme_gpu[01-02,09]
bmegpu_a10040g    up 5-00:00:00 1-infinite   no       NO        all      6       mixed bme_gpu[03-08]
bmegpu_a10040g    up 5-00:00:00 1-infinite   no       NO        all      1   allocated bme_gpu10
bmegpu_a10080g    up 5-00:00:00 1-infinite   no       NO        all      6       mixed bme_gpu[11-12,14-17]
bmegpu_a10080g    up 5-00:00:00 1-infinite   no       NO        all      1   allocated bme_gpu13
bme_fat           up 5-00:00:00 1-infinite   no       NO        all      1        idle bme_comput15
bme_quick         up    3:00:00 1-infinite   no       NO        all     18       mixed bme_comput[01,12,18],bme_gpu[01-09,11-12,14-17]
bme_quick         up    3:00:00 1-infinite   no       NO        all      3   allocated bme_comput16,bme_gpu[10,13]
bme_quick         up    3:00:00 1-infinite   no       NO        all     15        idle bme_comput[02-11,13-15,17,19]
bme_unlimited     up   infinite 1-infinite   no       NO        all      3       mixed bme_gpu[11-12,14]
bme_unlimited     up   infinite 1-infinite   no       NO        all      1   allocated bme_gpu13
bme_cpu*          up 10-00:00:0 1-infinite   no       NO        all      3       mixed bme_comput[01,12,18]
bme_cpu*          up 10-00:00:0 1-infinite   no       NO        all      1   allocated bme_comput16
bme_cpu*          up 10-00:00:0 1-infinite   no       NO        all     14        idle bme_comput[02-11,13-14,17,19]
[sunss@bme_login1 101203]$
```

查看详细队列信息：scontrol show partition

```
PartitionName=bme_cpu
   AllowGroups=ALL AllowAccounts=ALL AllowQos=ALL
   AllocNodes=ALL Default=YES QoS=partition_bme_cpu
   DefaultTime=12:00:00 DisableRootJobs=YES ExclusiveUser=NO GraceTime=0 Hidden=NO
   MaxNodes=UNLIMITED MaxTime=10-00:00:00 MinNodes=0 LLN=NO MaxCPUsPerNode=UNLIMITED
   Nodes=bme_comput[01-14,16-19]
   PriorityJobFactor=1 PriorityTier=6000 RootOnly=NO ReqResv=NO OverSubscribe=NO
   OverTimeLimit=NONE PreemptMode=OFF
   State=UP TotalCPUs=992 TotalNodes=18 SelectTypeParameters=NONE
   JobDefaults=(null)
   DefMemPerNode=UNLIMITED MaxMemPerNode=UNLIMITED
```

查看详细节点信息：scontrol show node 节点名

```
NodeName=bme_gpu17 Arch=x86_64 CoresPerSocket=24
   CPUAlloc=29 CPUTot=48 CPULoad=21.15
   AvailableFeatures=(null)
   ActiveFeatures=(null)
   Gres=gpu:NVIDIAA10080GBPCIe:4
   NodeAddr=bme_gpu17 NodeHostName=bme_gpu17 Version=19.05.5-1.0.0-29
   OS=Linux 3.10.0-957.el7.x86_64 #1 SMP Mon Dec 7 11:30:56 UTC 2020
   RealMemory=515470 AllocMem=278070 FreeMem=122918 Sockets=2 Boards=1
   MemSpecLimit=10240
   State=MIXED ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=bme_gpu,bmegpu_a10080g,bme_quick
   BootTime=2022-10-12T10:16:58 SlurmdStartTime=2022-10-12T10:20:39
   CfgTRES=cpu=48,mem=515470M,billing=48,gres/gpu=4
   AllocTRES=cpu=29,mem=278070M,gres/gpu=4
   CapWatts=n/a
   CurrentWatts=0 AveWatts=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

scontrol show node node-name | grep cpu #查看指定节点cpu状态
scontrol show node node-name | grep gpu #查看指定节点gpu状态

如何申请合适的资源

查看详细作业信息：scontrol show job

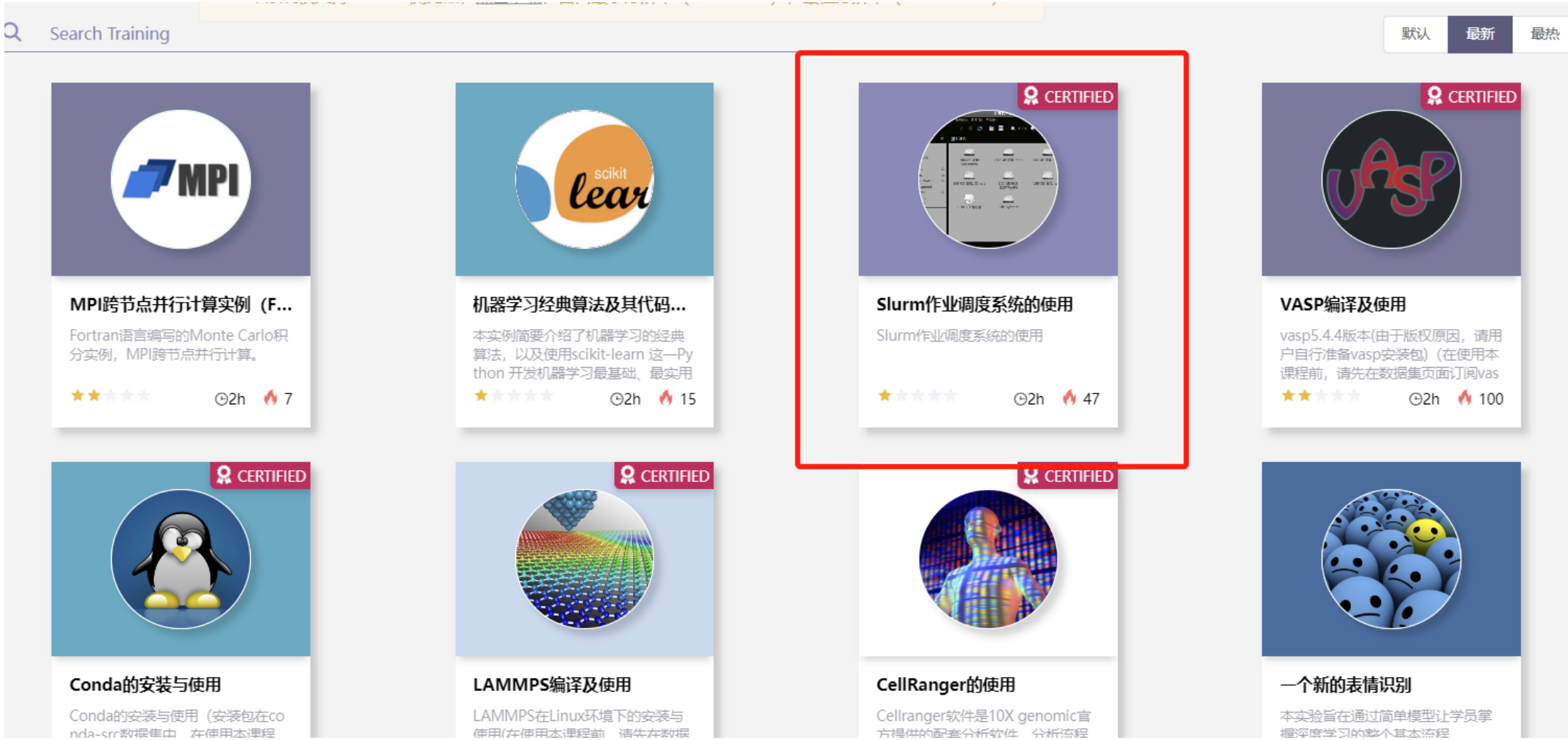
```
[        e_login1 slurm]$ scontrol show job
JobId=30401 JobName=test
   UserId=          4) GroupId=root(0) MCS_label=N/A
   Priority=1000 Nice=0 Account=sunss QOS=user_sunss
   JobState=PENDING Reason=Nodes_required_for_job_are_DOWN,_DRAINED_or_reserved_for_jobs_in_higher_priority_partitions Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=00:03:00 TimeMin=N/A
   SubmitTime=2022-10-20T13:45:03 EligibleTime=2022-10-20T13:45:03
   AccrueTime=2022-10-20T13:45:03
   StartTime=2022-10-21T22:07:15 EndTime=2022-10-21T22:10:15 Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2022-10-20T13:47:07
   Partition=bme_gpu AllocNode:Sid=bme_login1:120863
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=(null) SchedNodeList=bme_gpu16
   NumNodes=1-1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=1,mem=4414M,node=1,billing=1,gres/gpu=1
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryCPU=4414M MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/public/ho          /slurm/gpu.sh
   WorkDir=/public/hom          slurm
   StdErr=/public/home          urm/slurm-30401.out
   StdIn=/dev/null
   StdOut=/public/home/          urm/slurm-30401.out
   Power=
   TresPerNode=gpu:1
```

拓展学习

- 学校教学实训平台

拓展学习

- 学校教学实训平台



PBS和SLURM相关命令的对应关系

	PBS	SLURM
任务名称	#PBS -N name	#SBATCH -J name
指定队列/分区	#PBS -q node	#SBATCH -p node
最长运行时间	#PBS -l walltime=3:00:00	#SBATCH -t 3:00:00
指定节点数量	#PBS -l nodes=1	#SBATCH -N 1
指定CPU核心	#PBS -l ppn=56	#SBATCH -n=56
输出文件	#PBS -o test.out	#SBATCH -o test.out
提交任务脚本	qsub run.pbs	sbatch run.slurm
查看任务状态	qstat	squeue
取消任务	qdel job_id	scancel job_id