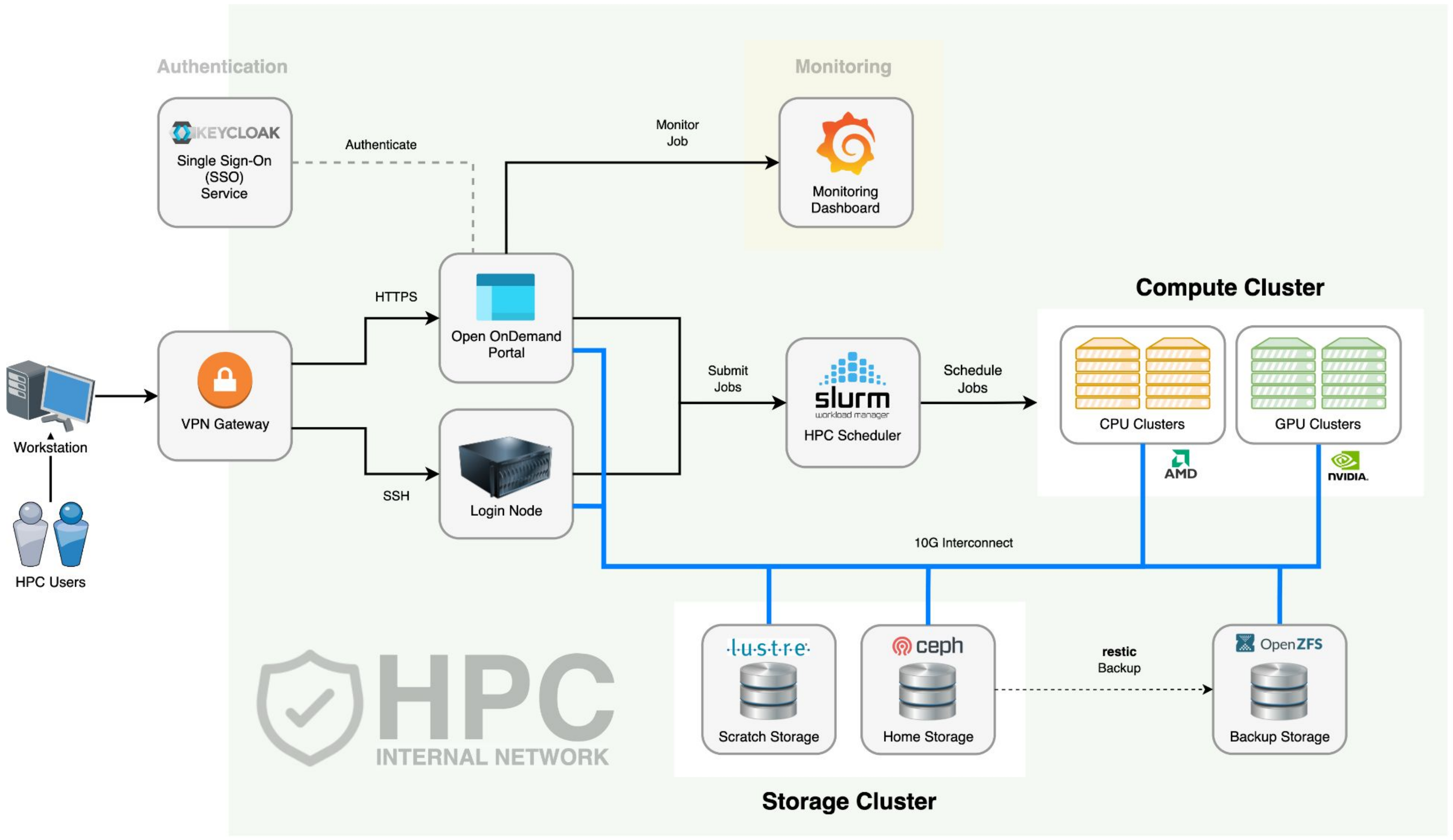
Basic Usage of HPC

Session Outcome

- Type of storages and differences
- Understand the components in UMHPC
- Understand the SLURM job parameters
- Understand how to submit a job
- How to check the log files

What is HPC

1



Authentication
KEYCLOAK
Single Sign-On (SSO) Service
Authenticate
Monitor Job
Monitoring
Monitoring Dashboard
HTTPS
Open OnDemand Portal
VPN Gateway
Workstation
HPC Users
SSH
Login Node
Submit Jobs
slurm workload manager
HPC Scheduler
Schedule Jobs
Compute Cluster
CPU Clusters
GPU Clusters
AMD
NVIDIA
10G Interconnect
lustre
Scratch Storage
ceph
Home Storage
restic Backup
OpenZFS
Backup Storage
Storage Cluster
HPC
INTERNAL NETWORK

Login Node

- The stuffs that users usually will do in here:
  - Transfer and manage files
  - Submit jobs
  - Check error and output logs
  - Monitor jobs
- Things to avoid:
  - Execute CPU or memory intensive scripts
  - Compile application
  - Extract large archive file

Storage Cluster

	Home Directory	Lustre Directory
File System	CephFS	Lustre
Directory	/home	/lustre
Storage Type	Persistent	Non-persistent
Storage Clean Up	No	Files that have not been accessed for 60 days
Quota	100 GB per user	Unlimited
Capacity	~ 87 TB	~ 230 TB
Backup	Daily	No
Group Directory	No	Yes

Compute Node

- Also referred as worker node.
- Some of the compute nodes have GPU.
- Your job will be executing in compute node(s).
- You do not have direct access to compute node unless you have submitted a job.

When Do You Need HPC?

- Your job will be running for very long period.
- Your job can utilize multiple CPU cores for parallelism.
- You need to repeat the same calculation with many different inputs.
- Your can utilize more powerful GPU(s).
- Your job consume very large amount of memory.

Basic Requirement to Access HPC

- A personal computing device
- Internet
- User account with HPC access
- OpenVPN and DICC OpenVPN profile

Account & Limits

- Every fresh user in DICC who wish to use HPC must request HPC access in DICC service desk.
- Every fresh HPC user will have limit resources access.

Resource	**Limited Account**	**Unlocked Account**
CPU	4	450
Memory	16 GB	2 TB
GPU	1	Unlimited
Wall Time	1 hour	7 days
QoS	limited	short, normal, long
Accessible Partition	cpu-opteron, gpu-k10, gpu-k40c	All partition

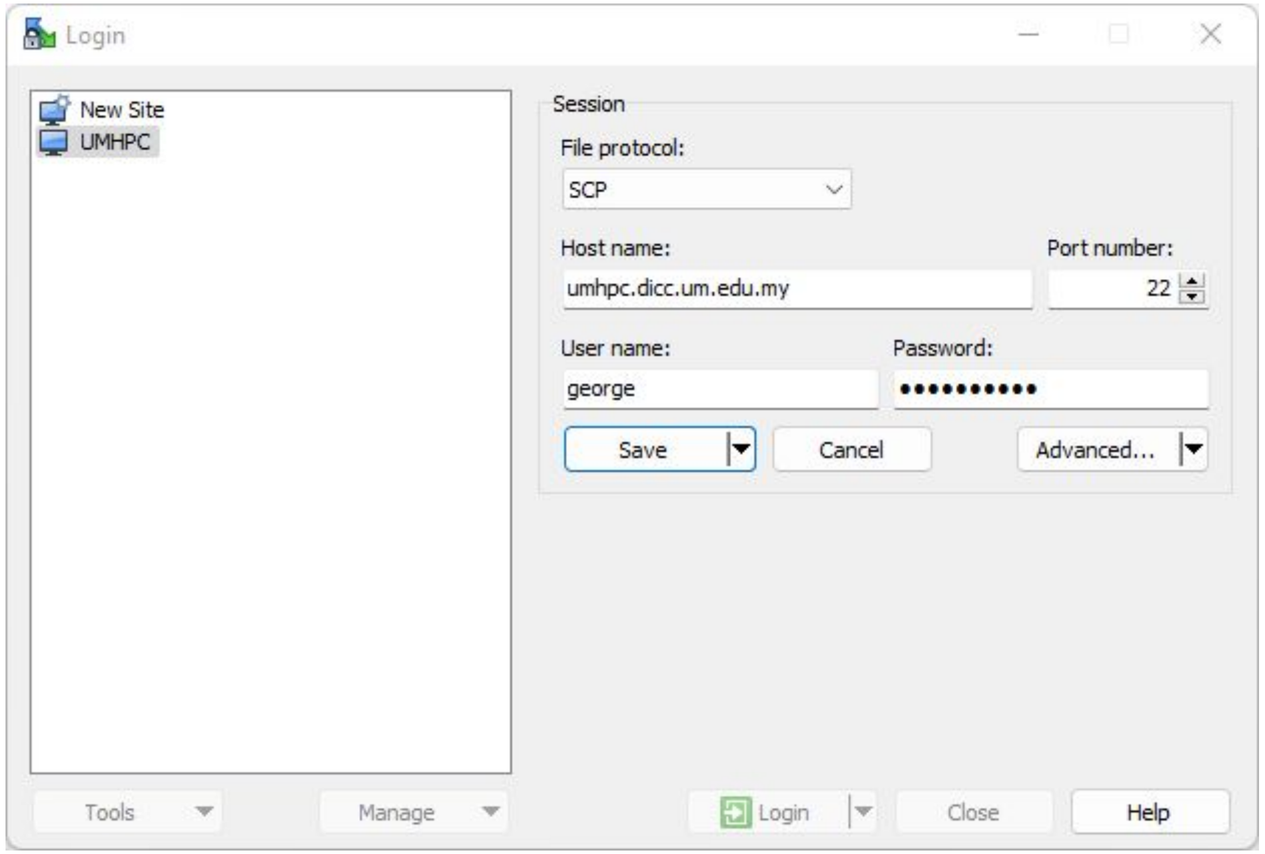
How to Submit A Job?

Things to Know

- File Transfer
- Application & Modules
- Job Parameter
- Job Submission
- Post-Job Submission

Transferring Files (Windows)

- For Windows user, we recommend user to use WinSCP:
  - Protocol: SCP
  - Port: 22
  - Host name: umhpc.dicc.um.edu.my



File protocol: SCP
Host name: umhpc.dicc.um.edu.my
Port number: 22
User name: george
Password: ●●●●●●●●●●●

Transferring Files (Linux/MacOS)

- You can use **FileZilla** as your FTP/SCP client to transfer your files between UMHPC and your local workstation.

Alternatively (For all OS)

- You can use **scp** command in your terminal/console/command prompt:
  - To transfer file into UMHPC:
    - `$ scp /path/to/filename username@umhpc.dicc.um.edu.my:/path/to/destination`
  - To transfer folder into UMHPC:
    - `$ scp -r /path/to/directory username@umhpc.dicc.um.edu.my:/path/to/destination`

Hands On

- Create a folder, **my_first_job** in your local machine.
- Create an empty text file, **tutorial.sh**
- Transfer the folder into **UMHPC**.

Things to Know

- File Transfer
- Application & Modules
- Job Parameter
- Job Submission
- Post-Job Submission

Application & Modules

- Most of the application/module or system library are NOT available in login node.

Function	Login Node	Compute node
List all applications in all compute nodes	`node-modules`	-
List all application in current instance	`module avail`	`module avail`
Load a specific application	`module load`	`module load`
List all the loaded application/module	`module list`	`module list`
Unload a loaded module	`module unload`	`module unload`
Unload all loaded module	`module purge`	`module purge`

Hands On

- Verify the presence of miniconda using the command:
  - **`conda --version`**
- Check the available module installed in login node.
- Load miniconda module.
- List all the module(s) had been loaded currently.
- Verify again the presence of miniconda using the command:
  - **`conda --version`**
- Unload all the modules.
- List out all the module installed in compute nodes.

Answer

```
$ conda --version
$ module avail
$ module load miniconda/conda-23.5.2
$ module list
$ conda --version
$ module purge
$ node-modules
```

Things to Know

- Application & Modules
- File Transfer
- Job Parameter
- Job Submission
- Post-Job Submission

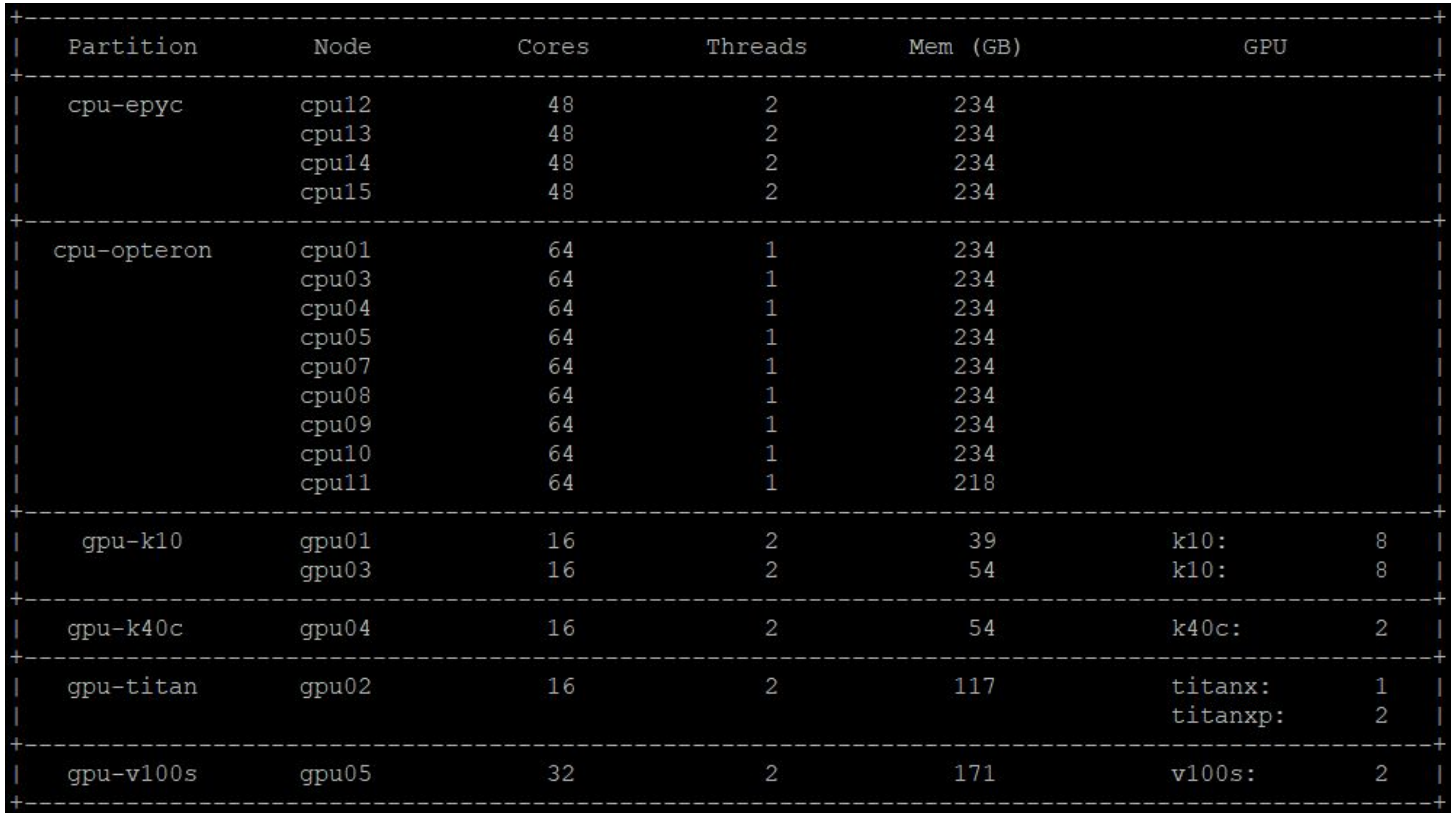
Job Parameters

- Job parameters determine what kind of resources you want.

Parameter	Description	Example
--partition, -p	Specify the partition to run job.	--partition=cpu-opteron
--ntasks, -n	Specify the number of CPUs/cores required.	--ntasks=4
--mem	Specify the amount of memory needed per node.	--mem=16G
--nodes, -N	Specify the number of compute nodes.	--nodes=1
--job-name, -J	Specify the name of the job.	--job-name=job01
--gpus, -G	Specify the number and the type of GPU card needed.	--gpus=1 or --gpus=titanxp:1
--qos, -q	Specify the QoS for the job	--qos=normal
--output, -o	Specify the filename for output log.	--output=/home/user/george/output.log
--error, -e	Specify the filename for error log.	--error=/home/user/george/error.log
--hint	Enable/Disable hyper-threading	--hint=nomultithread

Resources

- Resources summary can be displayed by using the command:
  - **`cluster-info`**



```
+-------------------------------------------------------------------------------------------+
|  Partition        Node          Cores        Threads      Mem (GB)              GPU       |
+-------------------------------------------------------------------------------------------+
|  cpu-epyc         cpu12          48             2           234                           |
|                   cpu13          48             2           234                           |
|                   cpu14          48             2           234                           |
|                   cpu15          48             2           234                           |
+-------------------------------------------------------------------------------------------+
| cpu-opteron       cpu01          64             1           234                           |
|                   cpu03          64             1           234                           |
|                   cpu04          64             1           234                           |
|                   cpu05          64             1           234                           |
|                   cpu07          64             1           234                           |
|                   cpu08          64             1           234                           |
|                   cpu09          64             1           234                           |
|                   cpu10          64             1           234                           |
|                   cpu11          64             1           218                           |
+-------------------------------------------------------------------------------------------+
|   gpu-k10         gpu01          16             2            39          k10:           8 |
|                   gpu03          16             2            54          k10:           8 |
+-------------------------------------------------------------------------------------------+
|  gpu-k40c         gpu04          16             2            54          k40c:          2 |
+-------------------------------------------------------------------------------------------+
|  gpu-titan        gpu02          16             2           117          titanx:        1 |
|                                                                          titanxp:       2 |
+-------------------------------------------------------------------------------------------+
|  gpu-v100s        gpu05          32             2           171          v100s:         2 |
+-------------------------------------------------------------------------------------------+
```

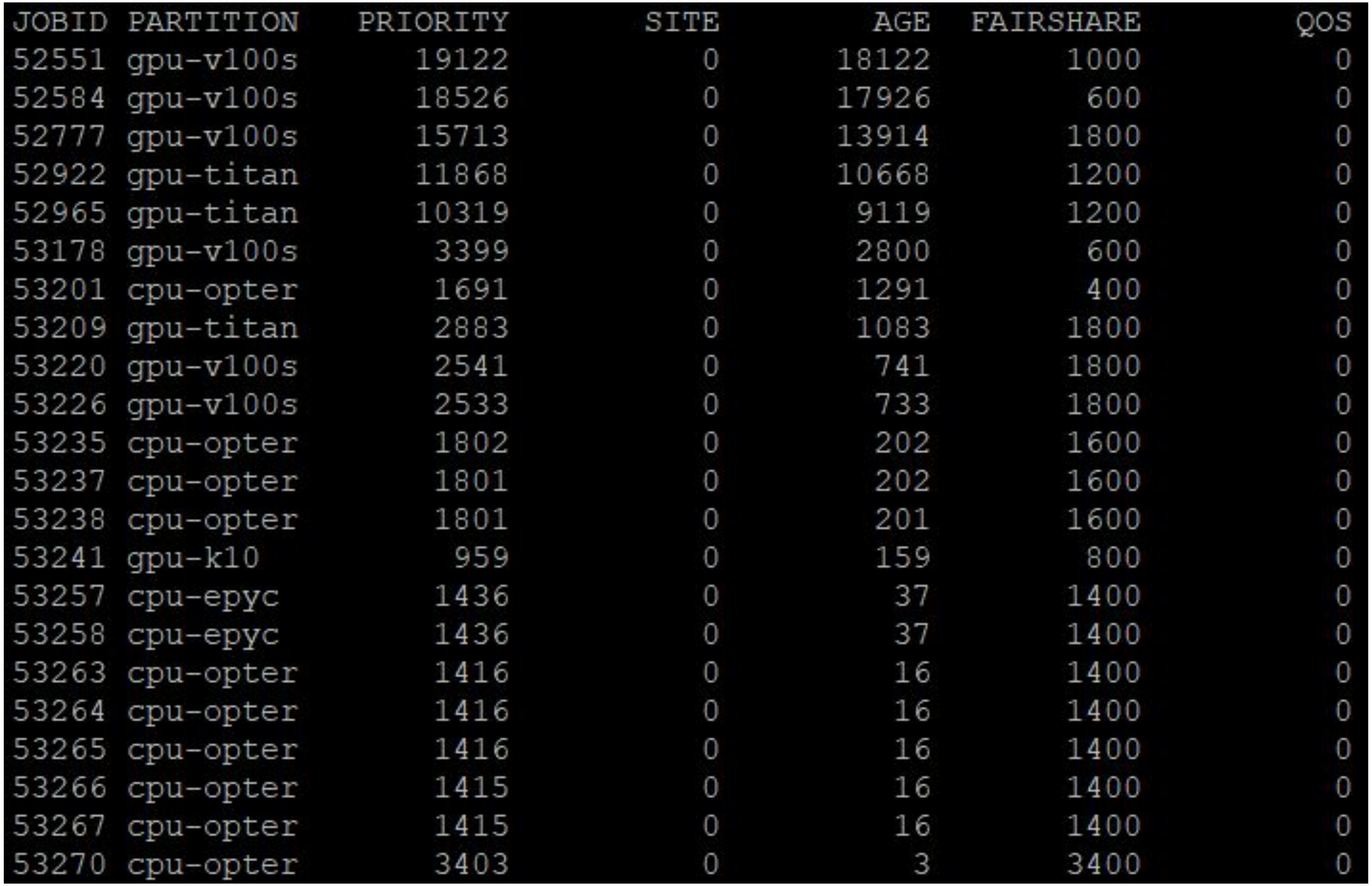
Partition

- Currently, there are 6 partitions available in DICC:
    - **`cpu-opteron`** (default) – 1800 MHz
    - **`cpu-epyc`** – 3200 MHz
    - **`gpu-k10`** – 745 MHz
    - **`gpu-k40c`** – 876 MHz
    - **`gpu-titan`**
        - **`Titan X`** – 1089 MHz
        - **`Titan Xp`** – 1582 MHz
    - **`gpu-v100s`** – 1597 MHz
- Do not submit CPU only job into GPU partition!

QoS

- QoS determine the maximum walltime of a job.
- Every job have its own unique priority.
    - **`sprio`** command can be used to check job priority in the queue currently.

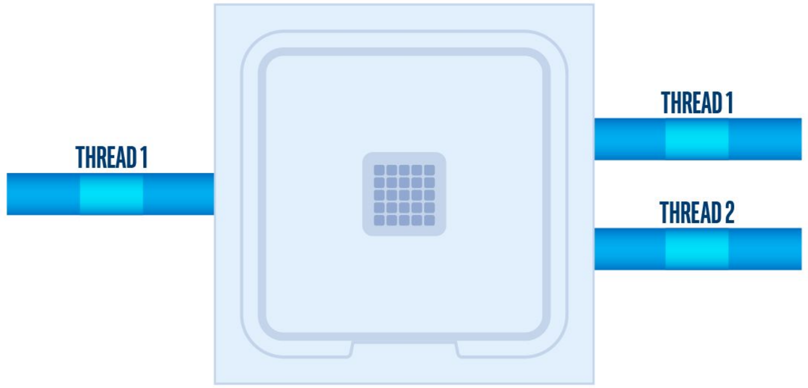
QoS	Maximum Wall time	Priority Boost
Short	1 hour	+10,000
Normal (default)	1 day	–
Long	7 days	–



```
JOBID PARTITION   PRIORITY        SITE        AGE  FAIRSHARE      QOS
52551 gpu-v100s      19122           0      18122       1000        0
52584 gpu-v100s      18526           0      17926        600        0
52777 gpu-v100s      15713           0      13914       1800        0
52922 gpu-titan      11868           0      10668       1200        0
52965 gpu-titan      10319           0       9119       1200        0
53178 gpu-v100s       3399           0       2800        600        0
53201 cpu-opter       1691           0       1291        400        0
53209 gpu-titan       2883           0       1083       1800        0
53220 gpu-v100s       2541           0        741       1800        0
53226 gpu-v100s       2533           0        733       1800        0
53235 cpu-opter       1802           0        202       1600        0
53237 cpu-opter       1801           0        202       1600        0
53238 cpu-opter       1801           0        201       1600        0
53241 gpu-k10          959           0        159        800        0
53257 cpu-epyc        1436           0         37       1400        0
53258 cpu-epyc        1436           0         37       1400        0
53263 cpu-opter       1416           0         16       1400        0
53264 cpu-opter       1416           0         16       1400        0
53265 cpu-opter       1416           0         16       1400        0
53266 cpu-opter       1415           0         16       1400        0
53267 cpu-opter       1415           0         16       1400        0
53270 cpu-opter       3403           0          3       3400        0
```

Hyper-Threading

- It is highly recommended to include the **`--hint`** parameter in the submission script.
- In most of the scenario, disabling hyper-threading will yield better performance.
- To disable hyper-threading,
  - **`--hint=nomultithread`**



Recommendations

- Always start small and scale larger when your are confident on how much you have understood your job.

CPU(s)	GPU(s)	Memory (GB)	QoS	Usage
4	–	16	short	Troubleshooting & debug
16	–	50	normal	Standard quick job
16	–	50	long	Standard small job
24 for EPYC 32 for Opteron	–	100	long	Standard medium job
48 for EPYC 64 for OPTERON	–	200	long	Standard large job
8	1	16	normal	Standard small GPU job

Things to Know

- Application & Modules
- File Transfer
- Job Parameter
- Job Submission
- Post-Job Submission

Job Submission

Batch Mode	Interactive Mode
Use **<u>submission script</u>** to execute.	Enter the node to execute.
Job continue to execute even if you have lost connection or your session terminated.	Job terminated on connection lost/terminated session.
Cannot make changes during the execution.	Able to make interactive input during the execution.
Usually done by using the command: **`sbatch`**	**`salloc`** to allocate resources. **`srun`** to join allocated resources and run calculation.
Execute until the maximum walltime.	
Must go through queue for resources allocation.	

Batch Mode

- When to use Batch Mode:
  - You have unstable network connection.
  - The application take a long time to complete.
  - No input needed during the process of calculation.
  - You need to run same calculation/simulation multiple times with different input files.
- This method is the recommended and standard way of running a job in HPC environment.
- Requirements:
  - Job script
  - Job parameters
  - Commands to execute
  - Input files

Example of Batch Script

```
#!/bin/bash -l

#SBATCH --partition=cpu-epyc
#SBATCH --job-name=job01
#SBATCH --nodes=1
#SBATCH --ntasks=24
#SBATCH --mem=100G
#SBATCH --qos=normal
#SBATCH --hint=multithread

module load myModule
app -i input.file -o output.file
```

Batch Mode (cont.)

- Use **`sbatch`** command to submit the job script.
  - **`$ sbatch batch_script.sh`**
- Use **`scancel`** command to cancel and remove the submitted job from queue. (Note: Once the job is cancelled, it cannot be recovered!)
  - **`$ scancel 12345`**

Hands On

- Edit the script, tutorial.sh to fulfil the following scenario:
  - Submitting partition: cpu-opteron
  - Total number of CPU cores: 16
  - Number of nodes: 2
  - Amount of memory per node: 50G
  - Quality of service: short
  - Job name: tutorial
  - Disabled hyper-threading

EXAMPLE

```
#!/bin/bash -l

#SBATCH --partition=cpu-epyc
#SBATCH --job-name=job01
#SBATCH --nodes=1
#SBATCH --ntasks=24
#SBATCH --mem=100G
#SBATCH --qos=normal
#SBATCH --hint=multithread

module load myModule
app -i input.file -o output.file
```

Answer

```
#!/bin/bash -l
#SBATCH --partition=cpu-opteron
#SBATCH --nodes=2
#SBATCH --ntasks=16
#SBATCH --mem=50G
#SBATCH --qos=short
#SBATCH --job-name=tutorial
#SBATCH --output=%x.out
#SBATCH --error=%x.err
#SBATCH --hint=nomultithread
```

Interactive Mode

- When to use Interactive Mode:
  - You have to input commands or intermediate input during the application execution.
  - You are trying to compile your own application.
  - You are trying to debug or troubleshoot your calculation or compilation.
- Requirements:
  - Job parameters
  - Commands to execute

Interactive Mode (cont.)

- To start an interactive session, first, you will need to allocate the resources you need then join the session interactively.
- To allocate resource for interactive session:
    - `$ salloc -p cpu-opteron -N 1 -n 4 --mem=16G --qos=normal`
- To join the allocated session interactively:
    - `$ srun --jobid=12345 --pty bash -l`
- To exit the interactive session, enter exit in terminal twice to leave and relinquish the allocated resources.

Example of Interactive Mode

```
[user@umhpc ~]$ salloc –p cpu-opteron –N 1 –n 4 --mem=16G --qos=normal
salloc: Pending job allocation 12345
salloc: job 12345 queued and waiting for resources
salloc: job 12345 has been allocated resources
salloc: Granted job allocation 12345
salloc: Waiting for resource configuration
salloc: Nodes cpu01 are ready for job
[user@umhpc ~]$ srun --jobid=12345 --pty bash –l
[user@cpu01 ~]$ exit
logout
[user@umhpc ~]$ exit
salloc: Relinquishing job allocation 12345
```

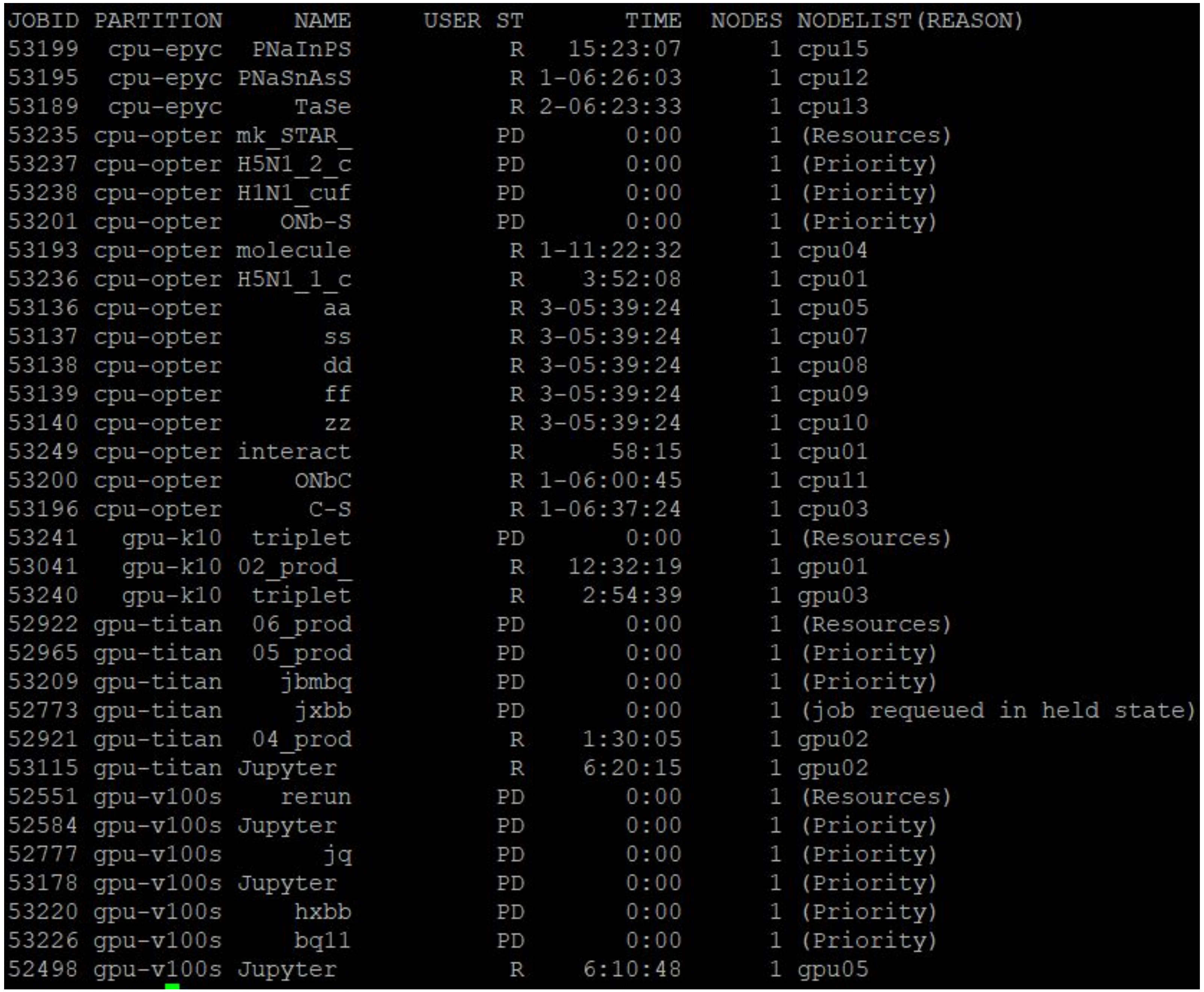
Things to Know

- Application & Modules
- File Transfer
- Job Parameter
- Job Submission
- Post-Job Submission

Job State

- You can check your own job(s) state with the command:
  - `$ squeue -u <username>`
- The common job state and the descriptions:

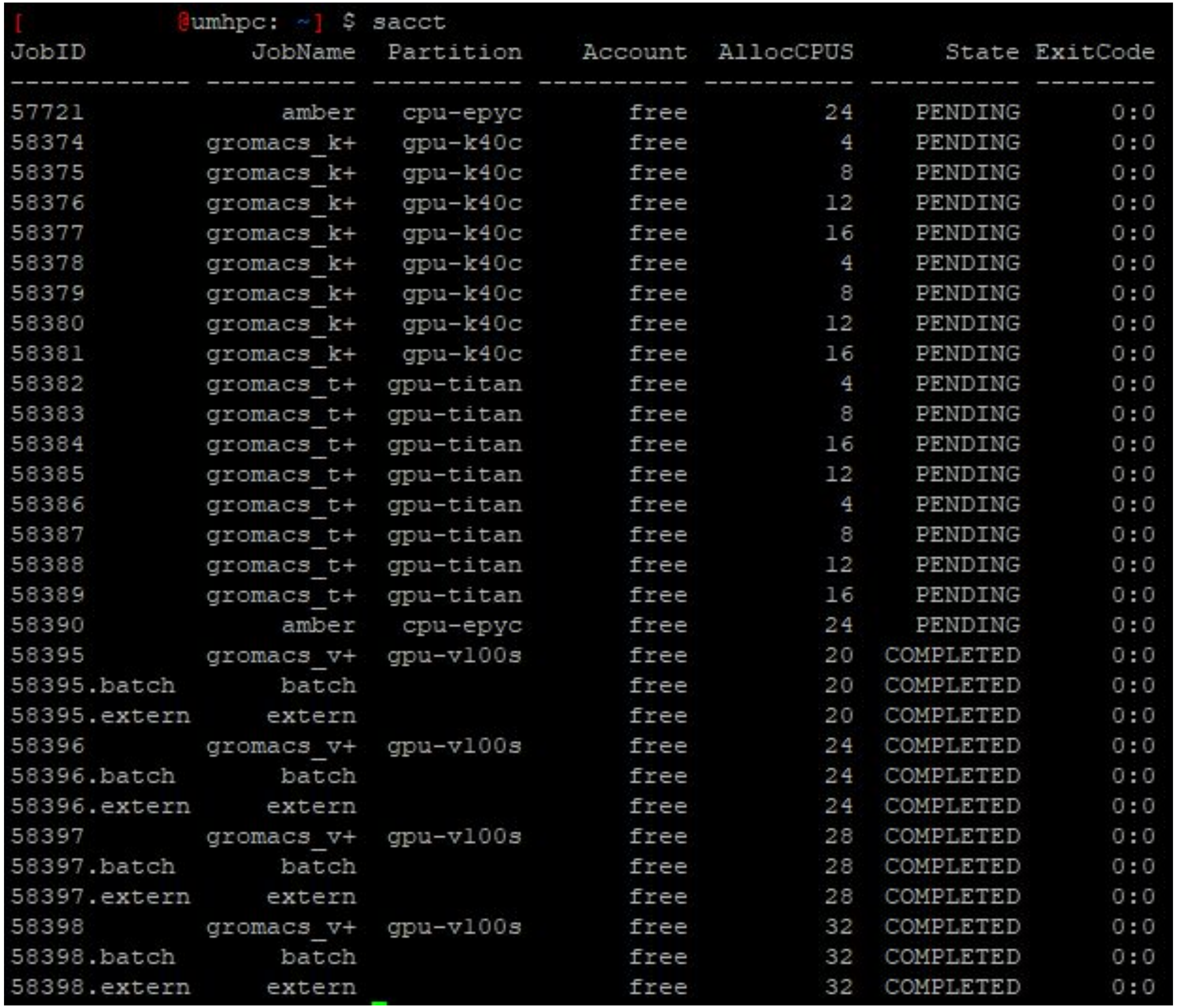
Job State	Description
PD / PENDING	Pending for resource scheduling.
R / RUNNING	The job is currently running.
RQ / REQUEUED	The job has been requeued.
CG / COMPLETING	The job has done execute and is now completing itself.
S / SUSPENDED	The job has been suspended.



```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
53199  cpu-epyc  PNaInPS           R   15:23:07      1 cpu15
53195  cpu-epyc PNaSnAsS           R 1-06:26:03      1 cpu12
53189  cpu-epyc     TaSe           R 2-06:23:33      1 cpu13
53235 cpu-opter mk_STAR_          PD       0:00      1 (Resources)
53237 cpu-opter H5N1_2_c          PD       0:00      1 (Priority)
53238 cpu-opter H1N1_cuf          PD       0:00      1 (Priority)
53201 cpu-opter    ONb-S          PD       0:00      1 (Priority)
53193 cpu-opter molecule           R 1-11:22:32      1 cpu04
53236 cpu-opter H5N1_1_c           R    3:52:08      1 cpu01
53136 cpu-opter       aa           R 3-05:39:24      1 cpu05
53137 cpu-opter       ss           R 3-05:39:24      1 cpu07
53138 cpu-opter       dd           R 3-05:39:24      1 cpu08
53139 cpu-opter       ff           R 3-05:39:24      1 cpu09
53140 cpu-opter       zz           R 3-05:39:24      1 cpu10
53249 cpu-opter interact           R      58:15      1 cpu01
53200 cpu-opter     ONbC           R 1-06:00:45      1 cpu11
53196 cpu-opter      C-S           R 1-06:37:24      1 cpu03
53241   gpu-k10  triplet          PD       0:00      1 (Resources)
53041   gpu-k10 02_prod_           R   12:32:19      1 gpu01
53240   gpu-k10  triplet           R    2:54:39      1 gpu03
52922 gpu-titan  06_prod          PD       0:00      1 (Resources)
52965 gpu-titan  05_prod          PD       0:00      1 (Priority)
53209 gpu-titan    jbmbq          PD       0:00      1 (Priority)
52773 gpu-titan     jxbb          PD       0:00      1 (job requeued in held state)
52921 gpu-titan  04_prod           R    1:30:05      1 gpu02
53115 gpu-titan Jupyter            R    6:20:15      1 gpu02
52551 gpu-v100s    rerun          PD       0:00      1 (Resources)
52584 gpu-v100s Jupyter           PD       0:00      1 (Priority)
52777 gpu-v100s       jq          PD       0:00      1 (Priority)
53178 gpu-v100s Jupyter           PD       0:00      1 (Priority)
53220 gpu-v100s     hxbb          PD       0:00      1 (Priority)
53226 gpu-v100s     bq11          PD       0:00      1 (Priority)
52498 gpu-v100s Jupyter            R    6:10:48      1 gpu05
```

Job State (cont.)

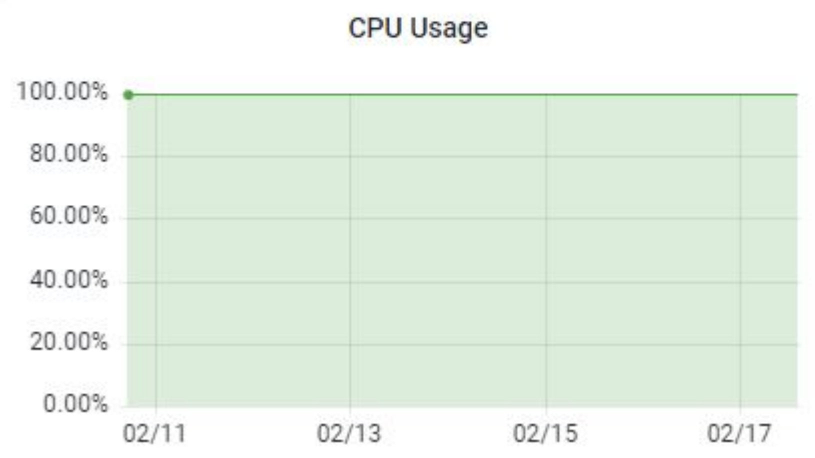
- If your job(s) is not in queue anymore, here are the possible scenario:
  - Your job(s) is/are completed. Check your output log for result.
  - Your job(s) is/are failed. Check your error log for error message.
- You can use **`sacct`** command to review the exit state of those jobs.
  - **`$ sacct --starttime=2023-01-01 --endtime=2023-01-31`**

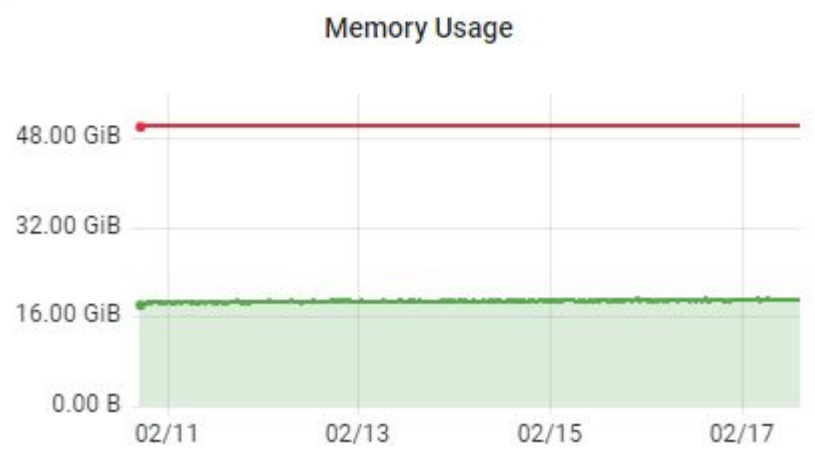


```
[[redacted]@umhpc: ~] $ sacct
JobID           JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
57721             amber   cpu-epyc       free         24    PENDING      0:0
58374        gromacs_k+   gpu-k40c       free          4    PENDING      0:0
58375        gromacs_k+   gpu-k40c       free          8    PENDING      0:0
58376        gromacs_k+   gpu-k40c       free         12    PENDING      0:0
58377        gromacs_k+   gpu-k40c       free         16    PENDING      0:0
58378        gromacs_k+   gpu-k40c       free          4    PENDING      0:0
58379        gromacs_k+   gpu-k40c       free          8    PENDING      0:0
58380        gromacs_k+   gpu-k40c       free         12    PENDING      0:0
58381        gromacs_k+   gpu-k40c       free         16    PENDING      0:0
58382        gromacs_t+  gpu-titan       free          4    PENDING      0:0
58383        gromacs_t+  gpu-titan       free          8    PENDING      0:0
58384        gromacs_t+  gpu-titan       free         16    PENDING      0:0
58385        gromacs_t+  gpu-titan       free         12    PENDING      0:0
58386        gromacs_t+  gpu-titan       free          4    PENDING      0:0
58387        gromacs_t+  gpu-titan       free          8    PENDING      0:0
58388        gromacs_t+  gpu-titan       free         12    PENDING      0:0
58389        gromacs_t+  gpu-titan       free         16    PENDING      0:0
58390             amber   cpu-epyc       free         24    PENDING      0:0
58395        gromacs_v+  gpu-v100s       free         20  COMPLETED      0:0
58395.batch       batch                  free         20  COMPLETED      0:0
58395.extern     extern                  free         20  COMPLETED      0:0
58396        gromacs_v+  gpu-v100s       free         24  COMPLETED      0:0
58396.batch       batch                  free         24  COMPLETED      0:0
58396.extern     extern                  free         24  COMPLETED      0:0
58397        gromacs_v+  gpu-v100s       free         28  COMPLETED      0:0
58397.batch       batch                  free         28  COMPLETED      0:0
58397.extern     extern                  free         28  COMPLETED      0:0
58398        gromacs_v+  gpu-v100s       free         32  COMPLETED      0:0
58398.batch       batch                  free         32  COMPLETED      0:0
58398.extern     extern                  free         32  COMPLETED      0:0
```

Job Monitoring

- Make sure your job(s) is running properly.
- Ways to monitor your job(s):
  - Visit DICC OnDemand portal at https://ood.dicc.um.edu.my/ under **Jobs** > **Active Jobs** section.
  - SSH into the node executing your jobs and use **`htop`** command for CPU usage and **`nvidia-smi`** for GPU usage.
  - Check your output log and error log.
- In most scenario, Opteron will yield 100% CPU usage and the other partition will yield a maximum of 50% CPU usage.





FAQ

- My job was failed and the error message show “OOM error”. Why?
  - Out-of-Memory (OOM) error is due to the application tend to use more memory than allocated memory. You try to allocate more memory for that particular job.
- Why my job(s) is/are queueing in the queue for very long period?
  - Check the reason in squeue:
    - Priority: There are more jobs with higher priority than your job(s).
    - Resources: Your job(s) is/are up next once the job(s) currently running in the partition have completed.

Useful Portal

- DICC Website – https://dicc.um.edu.my
- DICC Jira Service Desk – https://jira.dicc.um.edu.my
- DICC Documentation Confluence – https://confluence.dicc.um.edu.my

Hands On

- Create a job script, **first_job.sh** in the directory, **my_first_job** to fulfil the following scenario:
  - Submit to **`cpu-opteron`** partition.
  - Allocate 4 CPUs, 8 GB memory and 1 node
  - QoS: **limited**
  - Job name: **my_first_job**
  - With output and error log specified
- Commands to be executed by the job:

```
echo "This is my first job in $(hostname -s)"
sleep 10m
```

- Submit the job as batch mode.
- Use **`squeue`** to check the job state.
- Use **`scancel`** to cancel the job.
- Use **`sacct`** to check your account history.

```
#!/bin/bash -l

#SBATCH --partition=cpu-epyc
#SBATCH --job-name=job01
#SBATCH --nodes=1
#SBATCH --ntasks=24
#SBATCH --mem=100G
#SBATCH --qos=normal
#SBATCH -hint=nomultithread

module load myModule
app -i input.file -o output.file
```

Answer

```
#!/bin/bash -l

#SBATCH --partition=cpu-opteron
#SBATCH --job-name=my_first_job
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --mem=8G
#SBATCH --output=%x.out
#SBATCH --error=%x.err
#SBATCH --qos=limited

echo “This is my first job in $(hostname -s)”
sleep 10m
```

Answer

```
[user@umhpc ~]$ sbatch first_job.sh
[user@umhpc ~]$ squeue
[user@umhpc ~]$ scancel <job_id>
[user@umhpc ~]$ sacct
```

Thank You!