DICC

Basic HPC Usage

Serving the Nation. Impacting the World.

UNIVERSITI MALAYA

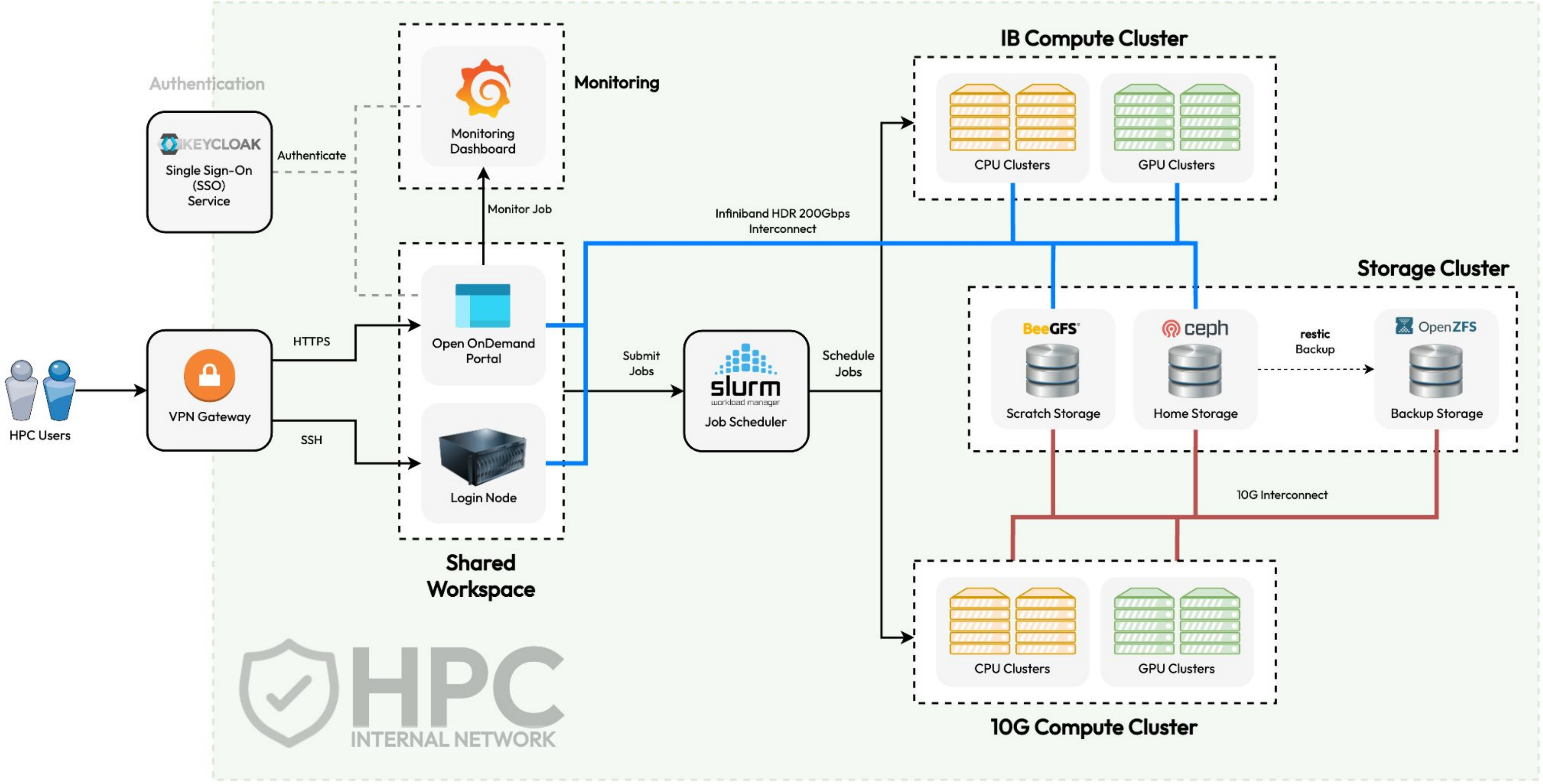
Session Outcome

- Understand the basic components of HPC.
- Understand the different storage and file system.
- Understand the basic SLURM parameters.
- Understand the concept of job submission.
- Understand the concept of job monitoring.

Basic Requirements for This Sessions

- Basic Linux knowledge
- DICC account with HPC access
- OpenVPN client
- DICC OpenVPN profile
- SSH client (PuTTY/MobaXterm/command prompt/terminal)
- WinSCP for Windows users; FileZilla for Linux/MacOS users.

UMHPC Architecture Design



Authentication

KEYCLOAK Single Sign-On (SSO) Service

Authenticate

Monitoring

Monitoring Dashboard

Monitor Job

HPC Users

VPN Gateway

HTTPS

SSH

Open OnDemand Portal

Login Node

Shared Workspace

Submit Jobs

slurm workload manager Job Scheduler

Schedule Jobs

Infiniband HDR 200Gbps Interconnect

IB Compute Cluster

CPU Clusters

GPU Clusters

Storage Cluster

BeeGFS Scratch Storage

ceph Home Storage

restic Backup

OpenZFS Backup Storage

10G Interconnect

10G Compute Cluster

CPU Clusters

GPU Clusters

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

ceph
Home Storage

BeeGFS
Scratch Storage

	Home Directory	Scratch Directory
Storage Solution	Ceph	BeeGFS
Directory	/home	/scr
Quota	100 GB per user	Unlimited
Raw Capacity	~303 TB	~ 466 TB
Storage Policy	Persistent	Non-persistent
Storage Cleanup Policy	No	Files that have not been accessed for 90 days or more.
Project Directory	No	Yes, /scr/project

Compute Node

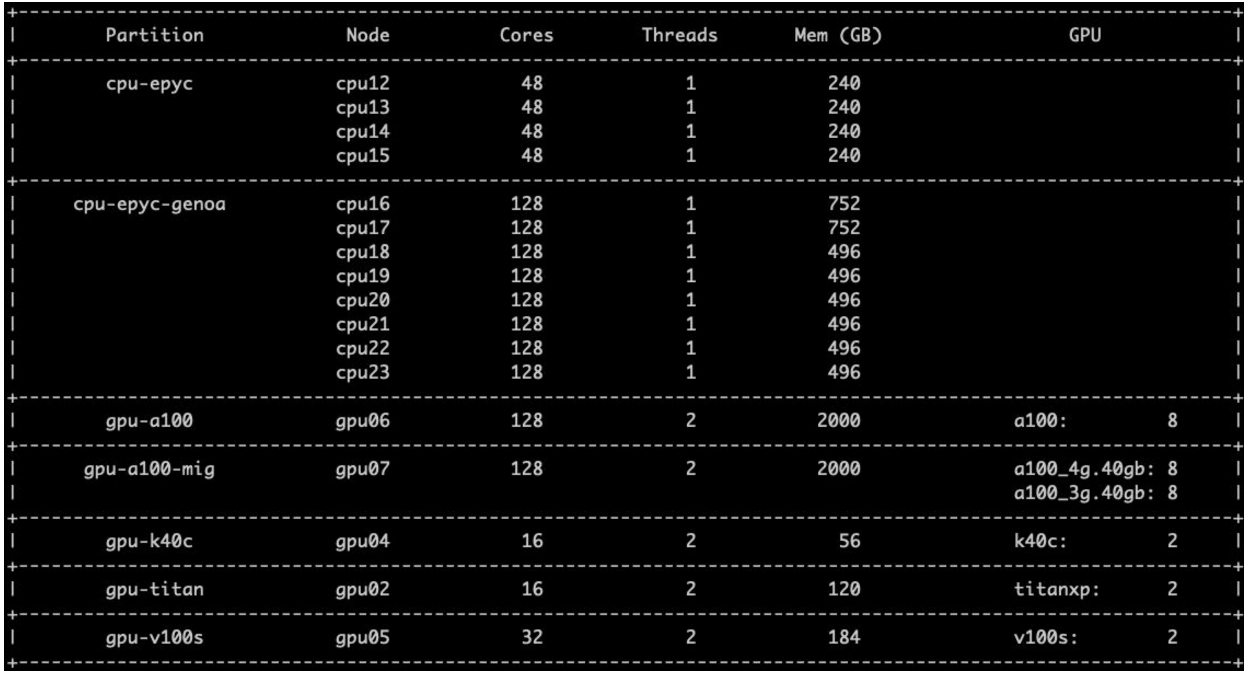
- Some compute nodes are attached with GPU card(s).
- All jobs must be submitted to be executed in **compute nodes** and **NOT login node**.
- You cannot access to compute nodes directly unless you have at least a job running in the compute node(s).

Compute Node (cont.)

- Currently, there are 7 partitions available in DICC:
  - cpu-epyc
    - AMD EPYC 7F72 24-Core Processor (3.2 GHz)
  - cpu-epyc-genoa
    - AMD EPYC 9534 64-Core Processor (3.7GHz)
  - gpu-k40c
    - Nvidia Tesla K40c - 3.5 GPU CC
  - gpu-titan
    - Nvidia Titan Xp - 6.1 GPU CC
  - gpu-v100s
    - Nvidia Tesla V100S - 7.0 GPU CC
  - gpu-a100
    - Nvidia A100 - 8.0 GPU CC
  - gpu-a100-mig
    - Nvidia A100 - 8.0 GPU CC

Compute Node (cont.)

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



```
+------------------------------------------------------------------------------------------------------------+
|          Partition            Node         Cores        Threads       Mem (GB)                GPU           |
+------------------------------------------------------------------------------------------------------------+
|          cpu-epyc             cpu12          48            1            240                                 |
|                               cpu13          48            1            240                                 |
|                               cpu14          48            1            240                                 |
|                               cpu15          48            1            240                                 |
+------------------------------------------------------------------------------------------------------------+
|       cpu-epyc-genoa          cpu16         128            1            752                                 |
|                               cpu17         128            1            752                                 |
|                               cpu18         128            1            496                                 |
|                               cpu19         128            1            496                                 |
|                               cpu20         128            1            496                                 |
|                               cpu21         128            1            496                                 |
|                               cpu22         128            1            496                                 |
|                               cpu23         128            1            496                                 |
+------------------------------------------------------------------------------------------------------------+
|          gpu-a100             gpu06         128            2           2000           a100:            8  |
+------------------------------------------------------------------------------------------------------------+
|        gpu-a100-mig           gpu07         128            2           2000           a100_4g.40gb:    8  |
|                                                                                       a100_3g.40gb:    8  |
+------------------------------------------------------------------------------------------------------------+
|          gpu-k40c             gpu04          16            2             56           k40c:            2  |
+------------------------------------------------------------------------------------------------------------+
|          gpu-titan            gpu02          16            2            120           titanxp:         2  |
+------------------------------------------------------------------------------------------------------------+
|          gpu-v100s            gpu05          32            2            184           v100s:           2  |
+------------------------------------------------------------------------------------------------------------+
```

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.

	Limited Account	Normal Account
Billing Limit	50,000	Unlimited
Accessible Partitions	cpu-epyc, gpu-k40c, gpu-titan, gpu-v100s	All partitions
Walltime	1 hour	7 days
QoS	limited	short, normal, long

Resource Usage

Priority

- Every job have unique priority.
- Priority determine which job will start first.
- Priority is determined by **job age**, **fairshare** and **QoS** in the ratio of 2:25:1.

Fairshare

- Fairshare is meant to maintain the fairness in queuing system.
- Every user have the same amount of initial fairshare.
- Fairshare is affected by the resource usage over the past 90 days.
- Resource usage is calculated by a billing system.

Billing System

- Every job submitted to compute node(s) will impose to a billing value.
- The billing value is calculated based on the cost of the node during acquisition.
- The billing amount for each resource type will be calculated using a ratio proportionally to the cost of the node, including CPUs, memory and GPUs.
- Each core allocated for non-multithreaded jobs will be treated as 2 CPUs and no multiple multithreaded jobs should fall within the same core.
- All jobs will be billed based on the **highest** amount of resource type allocated.

Billing System (cont.)

Partition	CPU	Memory	GPU	MaxPerNode
cpu-epyc	750	150	N/A	36000
cpu-epyc-genoa	625	120	N/A	80000
gpu-k40c	700	400	11200	22400
gpu-titan	750	200	12000	24000
gpu-v100s	1437.5	500	46000	92000
gpu-a100	4687.5	600	150000	1200000
gpu-a100-mig	4687.5	600	a100_4g.40gb: 83500 a100_3g.40gb: 66500	1200000

Example

- A non-multithreaded, 2 CPU cores, 64 GB memory and 2 v100s GPUs job running in gpu-v100s:
- The billing value can be breakdown as follow:
  - CPU = 4 (2 CPUs per core, 2 cores) * 1437.5 (Billing value per CPU in gpu-v100s) = **5750** resource usage per minute
  - Memory = 64 (64 GB memory) * 500 (Billing value per GB memory in gpu-v100s) = **32000** resource usage per minute
  - GPU = 2 (2 GPUs) * 46000 (Billing value per GPU in gpu-v100s) = **92000** resource usage per minute (**Highest**)
- Hence, the job will be billed for **92000** resource usage per minute as 2 v100s GPUs has the **highest** billing value per minute among 2 CPUs and 64 GB memory.

QoS

- QoS determine the maximum walltime, priority and resource usage factor of a job.

QoS	Priority	UsageFactor	Max WallTime
limited	0	10	1 hour
short	2000	1	1 hour
normal	0	1	1 day
long	0	1	7 days

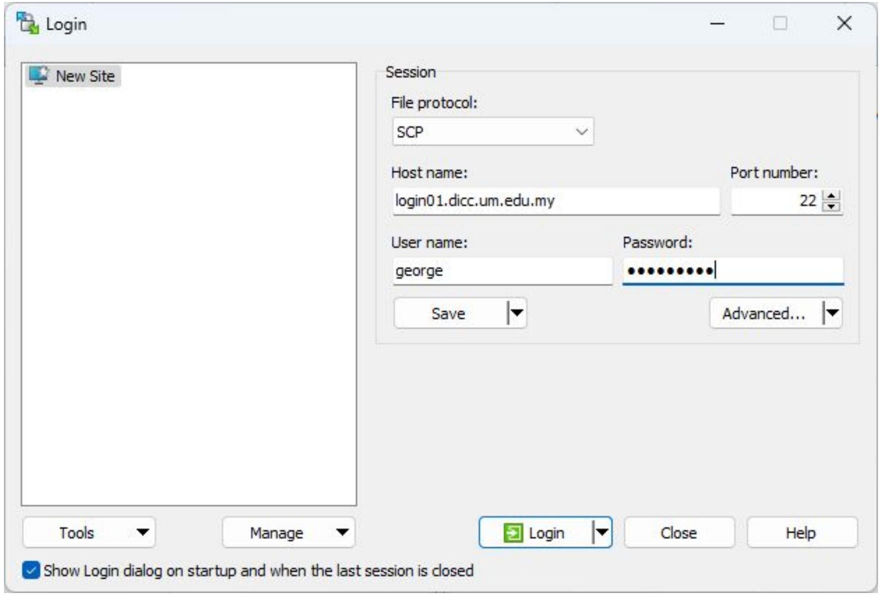
Basic SLURM Job Submission

Steps to Submit A Job

1. Prepare your input files.
2. Determine and load the application(s) of your choice.
3. Determine the SLURM job submission parameters.
4. Determine your job submission type.
5. Submit your job.

Prepare Your Input Files

- For Windows user, we recommend user to use **WinSCP**:
  - » Protocol: SCP
  - » Port: 22
  - » Host name: login01.dicc.um.edu.my



Login

New Site

Session

File protocol: SCP

Host name: login01.dicc.um.edu.my

Port number: 22

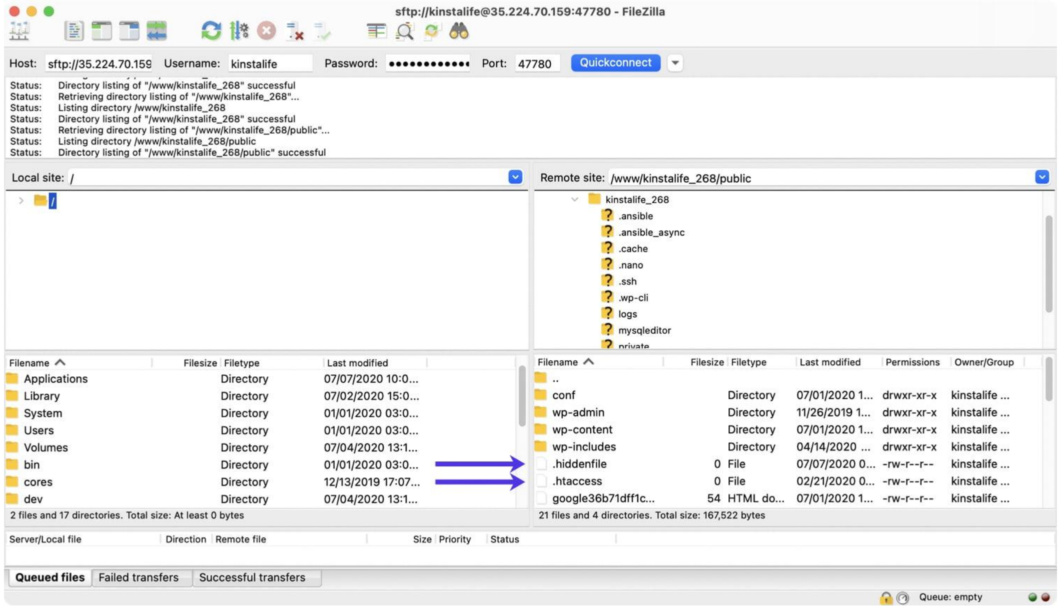
User name: george

Password: ●●●●●●●●●●

☑ Show Login dialog on startup and when the last session is closed

Prepare Your Input Files (cont.)

- For **Linux/MacOS**, you can use **FileZilla** as your FTP/SCP client to transfer your files between UMHPC and your local workstation.



sftp://kinstalife@35.224.70.159:47780 - FileZilla

Host: sftp://35.224.70.159 Username: kinstalife Password: ●●●●●●●●●●●● Port: 47780 Quickconnect

```
Status:     Directory listing of "/www/kinstalife_268" successful
Status:     Retrieving directory listing of "/www/kinstalife_268"...
Status:     Listing directory /www/kinstalife_268
Status:     Directory listing of "/www/kinstalife_268" successful
Status:     Retrieving directory listing of "/www/kinstalife_268/public"...
Status:     Listing directory /www/kinstalife_268/public
Status:     Directory listing of "/www/kinstalife_268/public" successful
```

Local site: /

Remote site: /www/kinstalife_268/public

```
kinstalife_268
    .ansible
    .ansible_async
    .cache
    .nano
    .ssh
    .wp-cli
    logs
    mysqleditor
    private
```

Filename	Filesize	Filetype	Last modified
Applications		Directory	07/07/2020 10:0...
Library		Directory	07/02/2020 15:0...
System		Directory	01/01/2020 03:0...
Users		Directory	01/01/2020 03:0...
Volumes		Directory	07/04/2020 13:1...
bin		Directory	01/01/2020 03:0...
cores		Directory	12/13/2019 17:07...
dev		Directory	07/04/2020 13:1...

2 files and 17 directories. Total size: At least 0 bytes

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..					
conf		Directory	07/01/2020 1...	drwxr-xr-x	kinstalife ...
wp-admin		Directory	11/26/2019 1...	drwxr-xr-x	kinstalife ...
wp-content		Directory	07/01/2020 1...	drwxr-xr-x	kinstalife ...
wp-includes		Directory	04/14/2020 ...	drwxr-xr-x	kinstalife ...
.hiddenfile	0	File	07/07/2020 0...	-rw-r--r--	kinstalife ...
.htaccess	0	File	02/21/2020 0...	-rw-r--r--	kinstalife ...
google36b71dff1c...	54	HTML do...	07/01/2020 1...	-rw-r--r--	kinstalife ...

21 files and 4 directories. Total size: 167,522 bytes

Server/Local file	Direction	Remote file	Size	Priority	Status

The HARDER Way To Transfer Files

- You can use `scp` command in your terminal/console/command prompt:
- To transfer file into UMHPC:

```
$ scp /path/to/filename username@login01.dicc.um.edu.my:/path/to/destination
```

- To transfer folder into UMHPC:

```
$ scp -r /path/to/directory username@login01.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 your home directory in UMHPC.

Steps to Submit A Job

1. Prepare your input files.
2. Determine and load the application(s) of your choice.
3. Determine the SLURM job submission parameters.
4. Determine your job submission type.
5. Submit your job.

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/24.1.2
$ module list
$ conda --version
$ module purge
$ node-modules
```

Steps to Submit A Job

1. Prepare your input files.
2. Determine and load the application(s) of your choice.
3. Determine the SLURM job submission parameters.
4. Determine your job submission type.
5. Submit your job.

SLURM Job Parameters

- Job parameters determine what kind of resources you want.

Parameter	Description	Example
--partition, -p	Specify the partition to run job.	--partition=cpu-epyc
--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 of GPU card needed.	--gpus=1

SLURM Job Parameters

Parameter	Description	Example
--qos, -q	Specify the QoS for the job	--qos=normal
--output, -o	Specify the filename for output log.	--output=output.log
--error, -e	Specify the filename for error log.	--error=error.log
--hint	Enable/Disable hyper-threading	--hint=nomultithread
--mail-type	Specify email notification on job status changes.	--mail-type=ALL
--mail-user	Specify which email address to receive the notification.	--mail-user=your_email@email.com

Steps to Submit A Job

1. Prepare your input files.
2. Determine and load the application(s) of your choice.
3. Determine the SLURM job submission parameters.
4. Determine your job submission type.
5. Submit your job.

SLURM Job Submission Mode

<table>
<tr><th>Batch Mode</th><th>Interactive Mode</th></tr>
<tr><td>Use <b><u>submission script</u></b> to execute.</td><td>Enter the node to execute (cloud-alike).</td></tr>
<tr><td>Job continue to execute even if you have lost connection or your session terminated.</td><td>Job terminated on connection lost/terminated session.</td></tr>
<tr><td>Cannot make changes during the execution.</td><td>Able to make interactive input during the execution.</td></tr>
<tr><td>Usually done by using the command:
<code>sbatch</code></td><td><code>salloc</code> to allocate resources.
<code>srun</code> to join allocated resources and run calculation.</td></tr>
<tr><td colspan="2">Execute until the maximum walltime.</td></tr>
<tr><td colspan="2">Must go through queue for resources allocation.</td></tr>
</table>

Steps to Submit A Job

1. Prepare your input files.
2. Determine and load the application(s) of your choice.
3. Determine the SLURM job submission parameters.
4. Determine your job submission type.
5. Submit your job.

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=nomultithread

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 <job id>
```

Hands On

Edit the script, tutorial.sh to fulfil the following scenario:

- Submitting partition: cpu-epyc
- Total number of CPU cores: 16
- Number of nodes: 2
- Amount of memory per node: 50 GB
- Quality of service: short
- Job name: tutorial

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-epyc
#SBATCH --nodes=2
#SBATCH --ntasks=16
#SBATCH --mem=50G
#SBATCH --qos=short
#SBATCH --job-name=tutorial
#SBATCH --output=%x.out
#SBATCH --error=%x.err
```

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-epyc -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@login01 ~]$ salloc -p cpu-epyc -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@login01 ~]$ srun --jobid=12345 --pty bash -l
[user@cpu12 ~]$ exit
logout
[user@login01 ~]$ exit
salloc: Relinquishing job allocation 12345
```

Basic SLURM Utilities

Job Queue Status

- You use `squeue` command to list all job in the current queue.
- To list your own job queue status:

```
$ squeue --me
```

- To check your job estimated starting time:

```
$ squeue --start --me
```

Job Status	Description
PD/Pending	Pending for resource scheduling.
R/Running	The job is currently running.

Job Priority

- You can use `sprio` command to list the priority of all current queuing jobs.
- The higher the number of job priority, the job is more likely to start next.

Job History

- You can use `sacct` command to review your account job history.
- To view your account history within a certain time frame:

```
$ sacct --starttime=2023-10-01 --endtime=2023-10-31
```

Job Monitoring

- Every user is **responsible** for monitoring your own jobs to prevent resource wastage.
- There are several options to monitor your job:
  - » 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.

Useful Portal

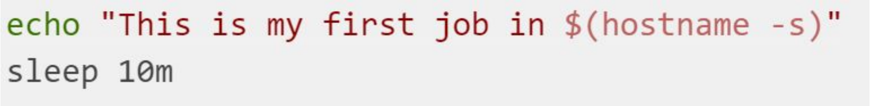
DICC Website – https://dicc.um.edu.my

DICC Jira Service Desk – https://jira.dicc.um.edu.my/servicedesk/customer/portals

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-epyc** partition.
  - Allocate 4 CPU cores, 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 slurm to check the job state.
- Use slurm to cancel the job.
- Use slurm to check your account history.

Example

```
#!/bin/bash -l

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

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

Answer

```
#!/bin/bash -l

#SBATCH --partition=cpu-epyc
#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
```

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

Thank You