Basic HPC Usage

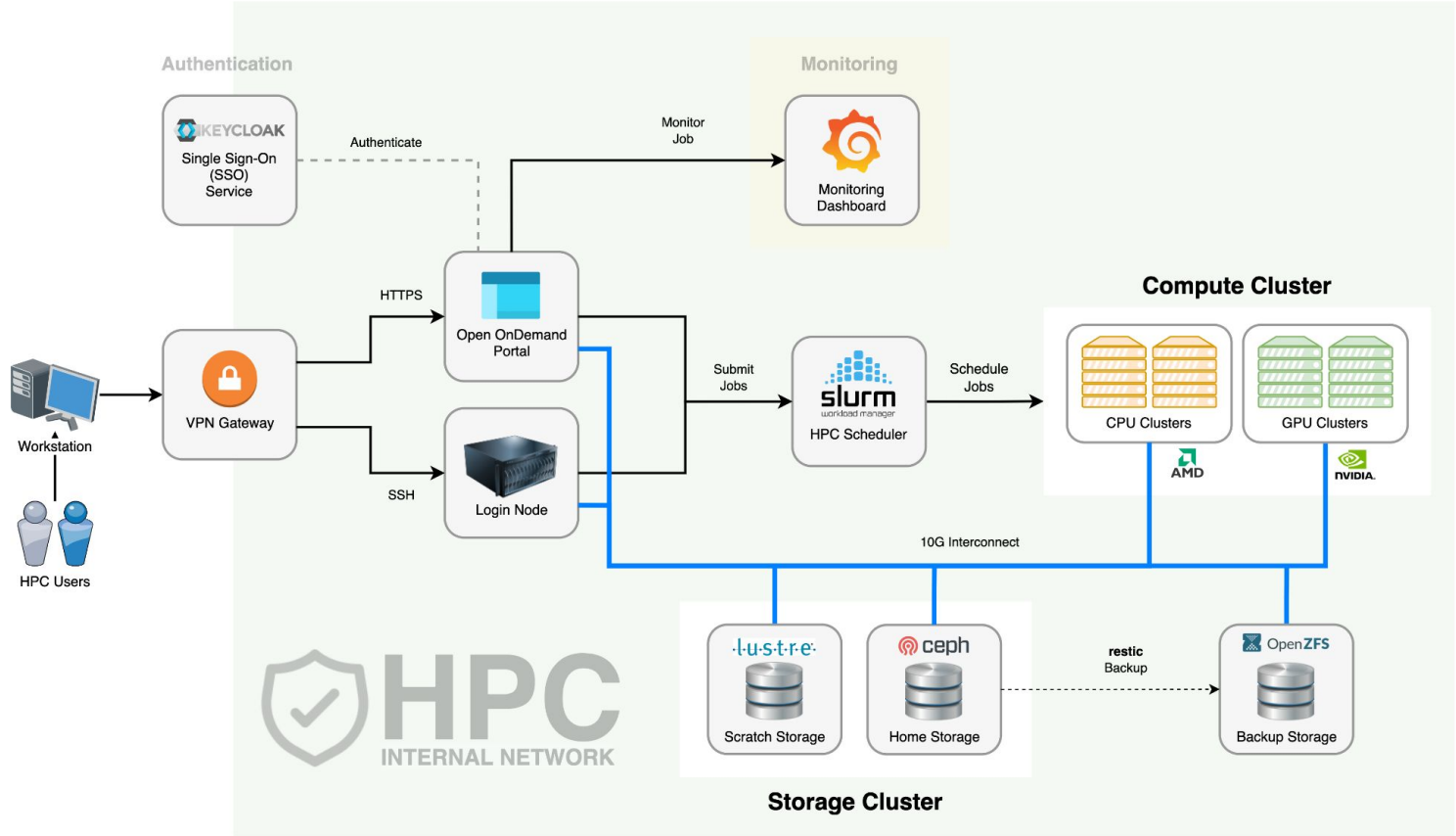
Session Outcome

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

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
Storage Solution	Ceph	Lustre
Directory	/home	/lustre
Quota	100 GB per user	Unlimited
Raw Capacity	~ 87 TB	~ 231 TB
Storage Policy	Persistent	Non-persistent
Storage Cleanup Policy	No	Files that have not been accessed for 60 days.
Project Directory	No	Yes, /lustre/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 6 partitions available in DICC:
  - cpu-opteron (default)
    - AMD Opteron Processor 6366 HE: 1800 MHz
  - cpu-epyc
    - AMD EPYC 7F72 24-Core Processor: 3200 MHz
  - gpu-k10
    - Nvidia Tesla K10 - 3.0 GPU Compute Capability (CC)
  - 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

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	2	240		
	cpu13	48	2	240		
	cpu14	48	2	240		
	cpu15	48	2	240		
cpu-opteron	cpu01	64	1	240		
	cpu03	64	1	240		
	cpu04	64	1	240		
	cpu05	64	1	240		
	cpu07	64	1	240		
	cpu08	64	1	240		
	cpu09	64	1	240		
	cpu10	64	1	240		
	cpu11	64	1	240		
gpu-k10	gpu01	16	2	32	k10:	8
	gpu03	16	2	56	k10:	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	12500	Unlimited
Accessible Partitions	cpu-opteron, gpu-k10, gpu-k40c	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-opteron	468.75	125	0	30000
cpu-epyc	375	150	0	36000
gpu-k10	656.25	375	2625	21000
gpu-k40c	700	400	11200	22400
gpu-titan	750	200	12000	24000
gpu-v100s	1437.5	500	46000	92000

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

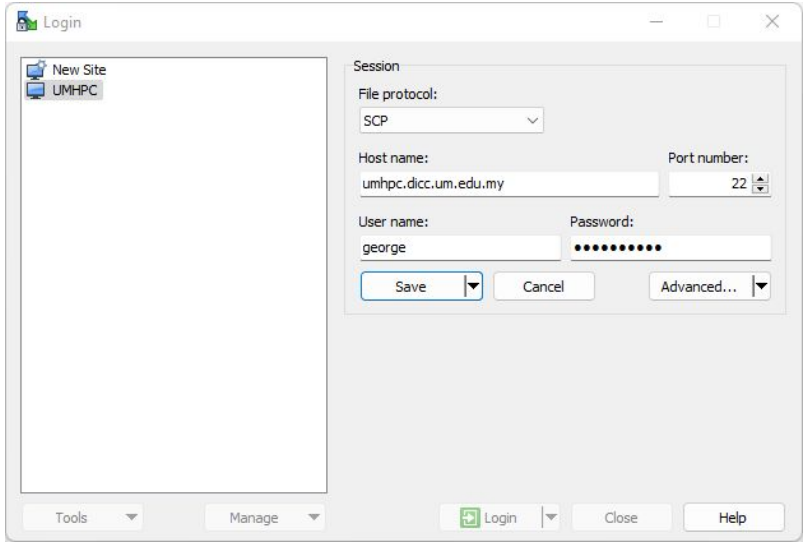
Basic SLURM Job Submission

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.

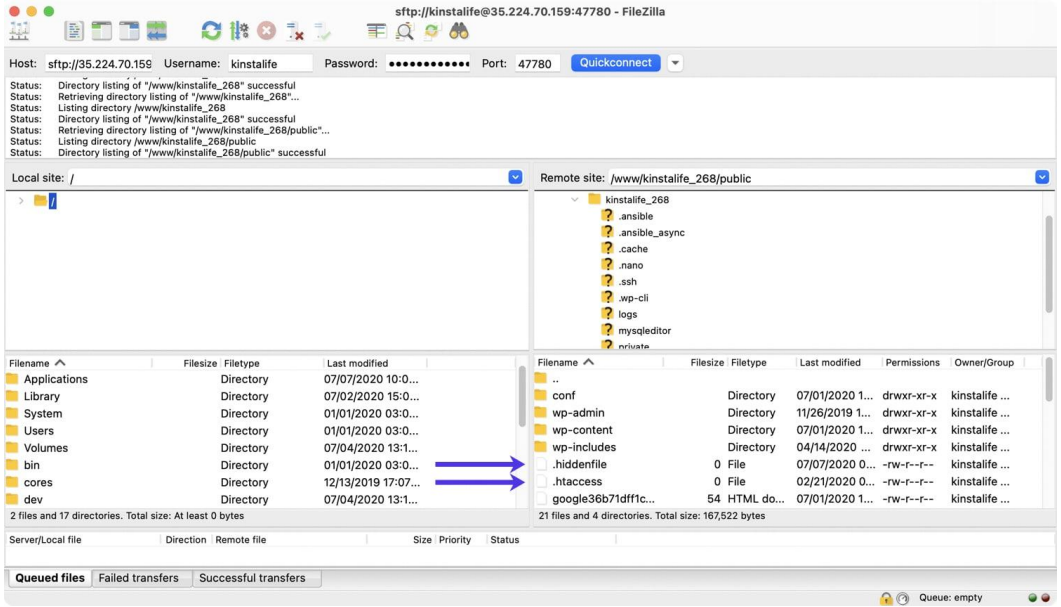
Prepare Your Input Files

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



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.



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@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 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/conda-23.5.2
$ 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.

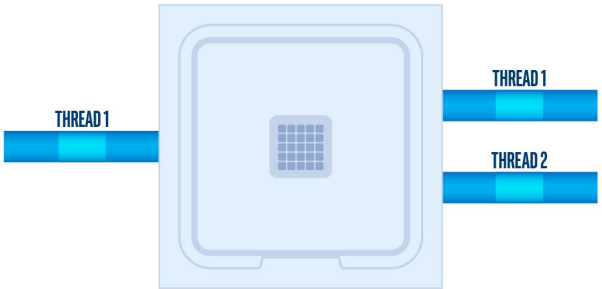
Parameter	Description	Example
--partition, -p	Specify the partition to run job.	--partition=cpu-opteron
--ntasks, -n	Specify the number of CPUs/cores required.	--ntasks=4
--mem	Specify the amount of memory needed per node.	--mem=16G
--nodes, -N	Specify the number of compute nodes.	--nodes=1
--job-name, -J	Specify the name of the job.	--job-name=job01
--gpus, -G	Specify the number 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

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`



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

Batch Mode	Interactive Mode
Use **submission script** to execute.	Enter the node to execute (cloud-alike).
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.	

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-opteron
- Total number of CPU cores: 16
- Number of nodes: 2
- Amount of memory per node: 50 GB
- 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
```

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 -u <your_username>
```

Job Status	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.

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.
- 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-opteron** 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 `squeue` to check the job state.
- Use `scancel` to cancel the job.
- Use `sacct` to check your account history.

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=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
```

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

Thank You