New Jersey Institute of Technology

High Performance Computing

An Introduction to NJIT High Performance Computing (HPC) and Services, Session II

April 17, 2024

Outline

- Summary of Session I
- Allocations
- User Environment
- Batch Processing
- Contact Us

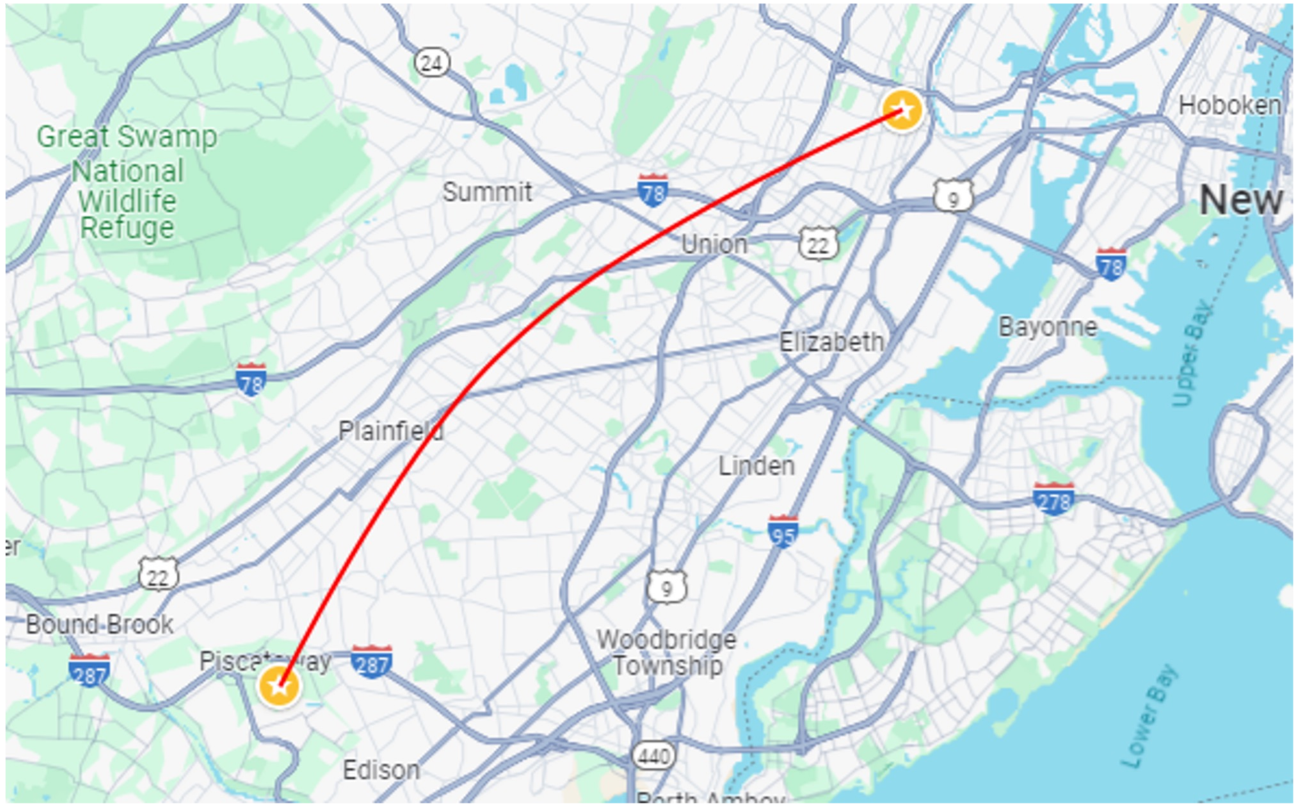
Summary of Session I

About NJIT HPC

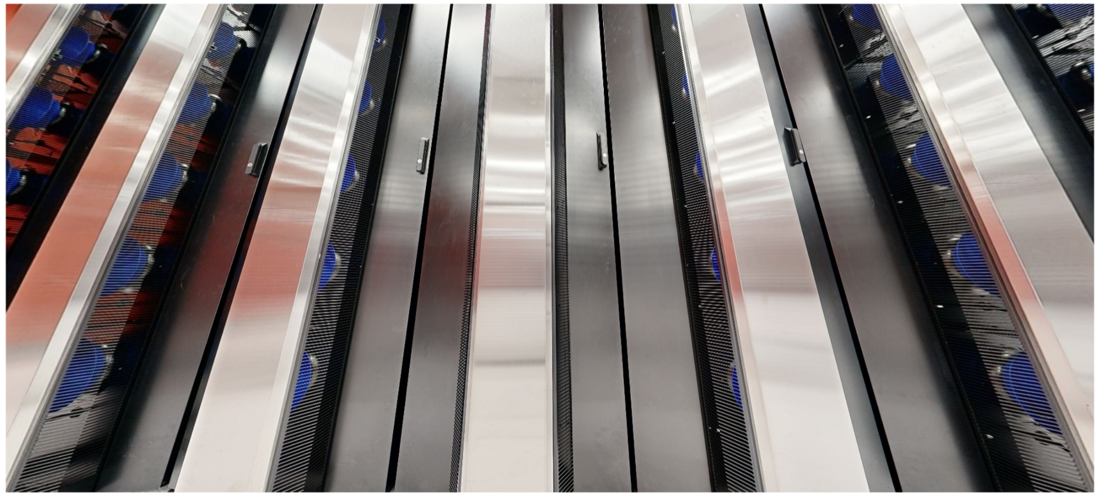
NJIT new high performance computing environment, built through a partnership with DataBank, a leading provider of enterprise-class colocation, connectivity and managed services, is live in DataBank's Piscataway, N.J. data center (EWR2) and will support NJIT's research efforts.

The services NJIT HPC provides

- High performance computing services
- Computational science expertise



Service Catalog



Cluster Computing

Built by Dell, the computing environment “Wulver” provides a total of 127 compute nodes or servers

Research Data Storage

High-performance, large capacity data storage spaces that are perfect for a wide variety of research data

Education

High performance computing and networking resources come together to create an exciting and innovative teaching and research environment

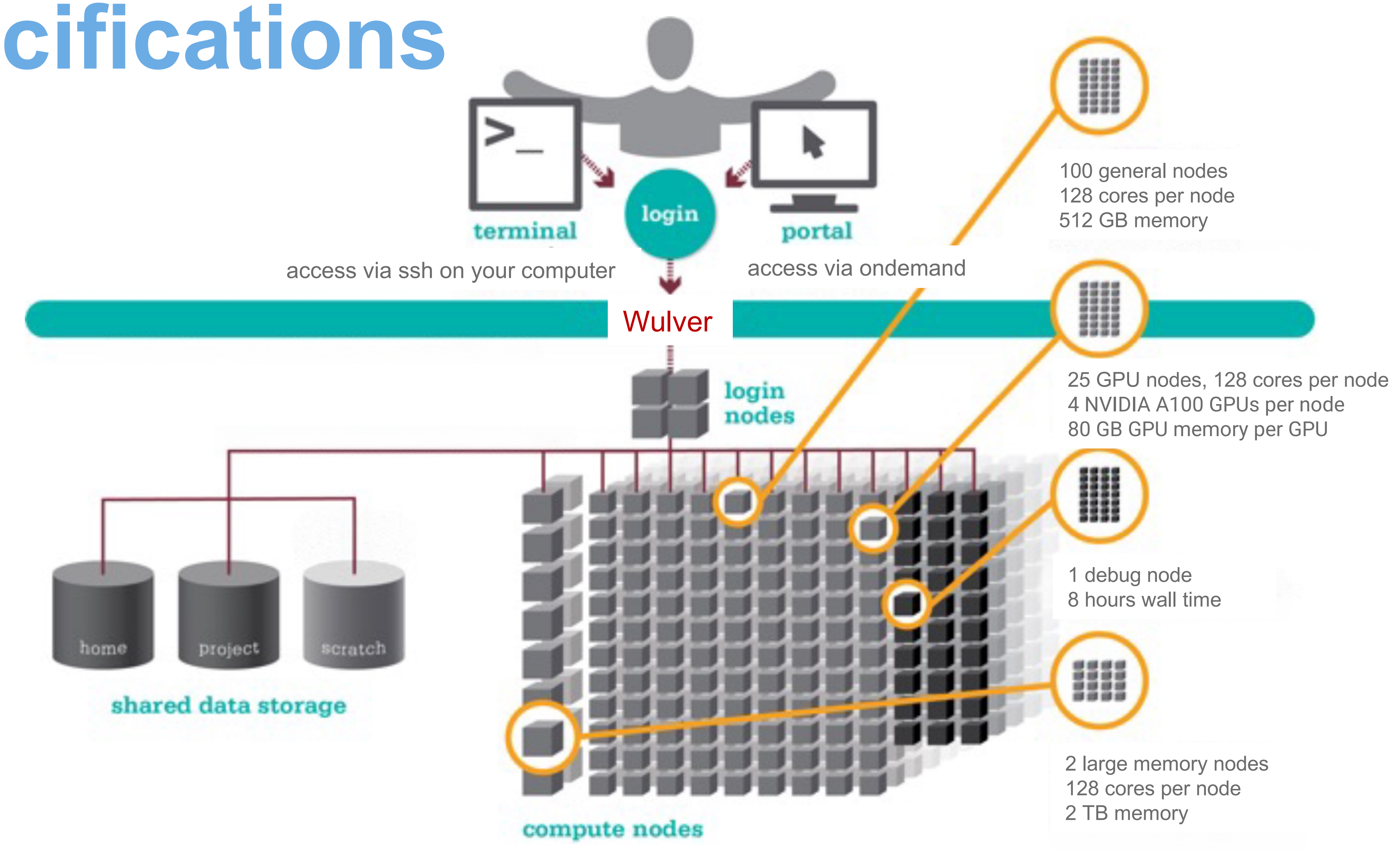
HPC Facilitation Service

Empowering users to perform essential research computing projects through training and effective user support

Scientific Software Development

Deep expertise in developing and deploying software

Wulver Cluster Specifications

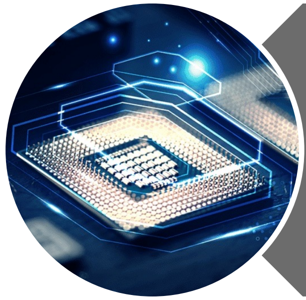


hpc.njit.edu/clusters/get_started_on_Wulver

HPC Allocations

HPC Allocations

- Accounting of computational resources for research
- Standard annual allocation – 300,000 SU’s per year
- Groups can purchase more if needed



Computing time

- Measured as CPU hours or SU (Service Units)
- 1 SU = Number of CPUs x Walltime in hours x usage factor

Storage

- Home (~50GB/user)
- Project (2TB/PI Group)
- Scratch (10TB/PI Group)

HPC Allocations - Storage

Filesystem	Purpose	Quota	Backed-Up?	Purged?
Home ($HOME)	Non-research such as profile, history	50GB	Yes, daily	No
Project (/project/$PI_UCID/ $LOGIN/)	Active research by groups.	2 TB/ PI Group	Yes, daily	No
Scratch (/scratch/PI_UCID/$ LOGIN)	Temporary space for intermediate results, downloads, checkpoints, and such. MOVE YOUR RESULTS & IMPORTANT FILES TO /project	10 TB/ PI Group	No	Yes – 30 days
Compute (/tmp)	Very high speed temporary storage	Varies (~1 TB)	No	Yes – after job ends

https://hpc.njit.edu/clusters/get_started_on_Wulver/#wulver-filesystems

NJIT

HPC Allocations - SU

- Example of SU charges: (20 cores with 4 GPU for 8 hours)
- SU = 20 x 8 x 3 = 320

Partition	Nodes	Cores /Node	CPU	GPU	Memory	SU charge
`--partition=general`	100	128	2.5G GHz AMD EPYC 7763 (2)	NA	512 GB	1 SU per hour per cpu
`--partition=debug`	1	4	2.5G GHz AMD EPYC 7763 (2)	NA	512 GB	No charges, must be used with `--qos=debug`
`--partition=gpu`	25	128	2.0 GHz AMD EPYC 7713 (2)	NVIDIA A100 GPUs (4)	512 GB	3 SU per hour per cpu
`--partition=bigmem`	2	128	2.5G GHz AMD EPYC 7763 (2)	NA	2 TB	1.5 SU per hour per cpu

https://hpc.njit.edu/Software/slurm/slurm/

NJIT

SU Charges

- Standard Priority (`--qos=standard`)
  - Faculty PIs are allocated 300,000 Service Units (SU) per year on request at no cost
  - Additional SUs may be purchased at a cost of $0.005/SU.
  - The minimum purchase is 50,000 SU ($250)
  - Wall time maximum - 72 hours
- Low Priority (`--qos=low`)
  - Not charged against SU allocation
  - Wall time maximum - 72 hours
  - Jobs can be preempted by those with higher and standard priority jobs when they are in the queue
- High Priority (`--qos=high`)
  - Not charged against SU allocation
  - Wall time maximum - 72 hours – can be increased based on PI’s request
  - Only available to contributors

User Environment

Connecting to Wulver

- Connect to Wulver using **`ssh`** (secure shell)
  - From a Linux/UNIX (and Mac) terminal: At prompt, enter

    `ssh` ***`ucid`*****`@wulver.njit.edu`** (Repalce “ucid” with NJIT UCID)

    `ssh -X -Y ucid`**`@wulver.njit.edu`** (For visualization)
  - From Windows: [**`Download`**]() **`MobaXterm`**

https://hpc.njit.edu/clusters/cluster_access

Transferring Files to and from the Cluster

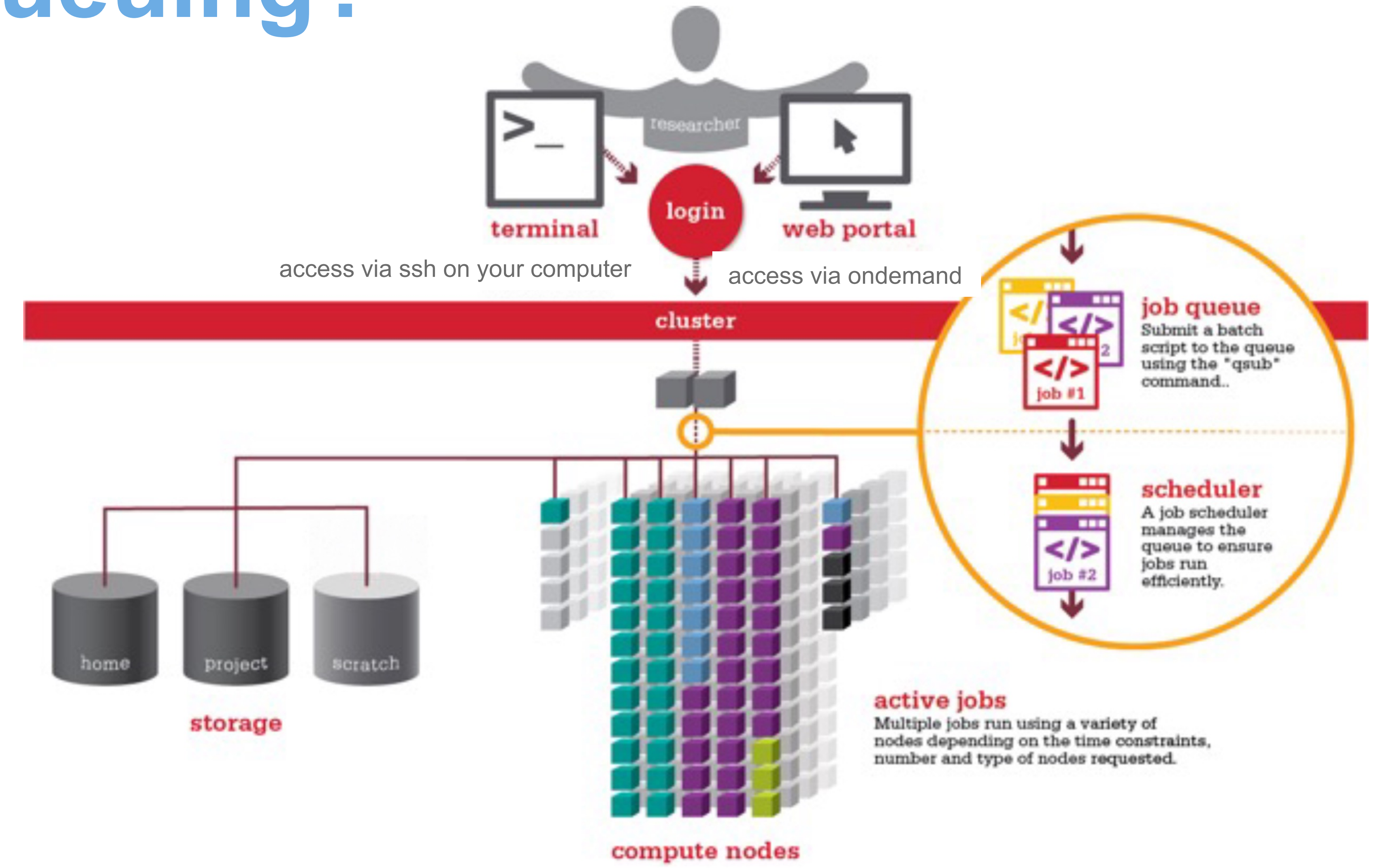
- `scp` (Secure Copy Protocol) is used to securely copy files/folders between Linux (Unix) systems on a network
  - `scp [option] [ucid@wulver.njit.edu:path/to/source/file] [target/path]` ## copy files from remote machine to local machine
  - `scp [option] [path/to/source/file] [ucid@wulver.njit.edu:target/path]` ## copy files from local machine to remote machine
- Example of `scp`
  - `scp -r example ucid@wulver.njit.edu:/home/dir` ## copy the "example" folder recursively to `/home/dir` on Lochness

Transferring Files to and from the Cluster

- `rsync` command also transfers and synchronizes files or directories between local and remote machine.
- `rsync` command syntax
- `rsync [optional modifiers] [SRC] [DEST]`
- Example of `rsync`
  - `rsync -azP ucid@wulver.njit.edu:/home/dir/* /home/work` ## copy all files and subdirectories under "dir" folder of Wulver recursively to /home/work of local machine
  - `rsync -azP --exclude=file1,file2  [SOURCE] [DEST]` ## exclude specific files or sub-directories
  - `rsync -azP /{,'company*/'{,'**'}} --exclude='*' [SOURCE] [DEST]` ## only copy files from subdirectories with name "company" (e.g., company1, company2, .. etc.) and exclude everything else

Batch Processing

Why do supercomputers use queuing?


researcher
terminal
login
web portal
access via ssh on your computer
access via ondemand
cluster
job queue
Submit a batch script to the queue using the "qsub" command..
job #1
scheduler
A job scheduler manages the queue to ensure jobs run efficiently.
job #2
home
project
scratch
storage
active jobs
Multiple jobs run using a variety of nodes depending on the time constraints, number and type of nodes requested.
compute nodes

Steps for Running a Job on the Compute Nodes

1. Create a batch script for a job
2. Prepare and gather input files in your directory
3. Submit the job
4. Job gets queued
5. Job runs when resources become available
6. Get your results in your directory when the job finishes

Specifying Resources in a Job Script

- Nodes and cores (processors) per node, GPUs
- Partition: CPU or GPU jobs, use `--partition`
- QoS, use `--qos`
- Account, use `--account=PI_ucid` # Replace PI_ucid with the UCID of PI
- Walltime, use `--time`
  - Maximum allowable walltime – 3 days
  - Shorter job may start sooner due to backfill
- Loading modules

Environment Modules

Environment Modules allows for dynamic modification and management of a user's environment via **modulefiles.**

- Manages multiple versions of software that require unique environments.
- Allows the user to load only the environment variables important to their applications, from within their job.

What modules do you have loaded?	**module list**
What modules are available?	**module spider** or **module avail**
Multiple versions of the same compiler	**module avail intel**
Add a software module to your environment	**module load CUDA**
Remove a software package from your environment	**module unload intel**
Load a different software version	**module swap intel intel/2021b**

General Application Workflow

- Log into cluster as your user.
- Copy input files to new directory.
- Change to copied directory via command line.

```
cd /path/to/copied_directory
```

- Copy job a template to the directory.

```
cp /path/to/templates/jobtemplate.job jobfile.job
```

- Modify the job file:
  - Change the number of resources to desired number.
  - Change the module load command based on the application name and version.
  - Update command line with commands required for job.
- Submit the job file using ‘sbatch’.

Manage Jobs – Options

Reporting Options

Directive	Options	Description
--error	File	Define standard error file
--out	File	Define standard output file
--job	Name	Define job name
--mail-type=	ALL, BEGIN, END, FAIL, REQUEUE	Notify user by email when <type> event occurs
--mail-user=	Email address	Send email to this address for events specified with mail-type option (default is submitting user).

Limit Options

Directive	Options	Description
--ntasks	Number of cpus	Number of CPUs (tasks) to be allocated
--nodes	Node	Number of Nodes to be allocated
--partition	Partition	Request a partition of resources for job allocation (queue)
--time	Time [[d-]h:]m[:s]	Minimum time limit on job allocation

see https://slurm.schedmd.com/srun.html for more details

Sample Simple Job Script

```
#!/bin/bash

#SBATCH --job-name=my_job
#SBATCH --partition=general
#SBATCH --output=%x.%j.out
#SBATCH --error=%x.%j.err
#SBATCH --account=PI_UCID
#SBATCH --qos=low
#SBATCH --time=00:20:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mail-type=ALL
#SBATCH –mail-user=ab1234@njit.edu

date
sleep 60
date
```

Job setup information for SLURM

Commands to be run

- This runs a batch job called “my_job” to the “general” partition, with 1 task and 4 cpus per task, a wall time limit of 20 minutes.
- QOS is required. Account is recommended.
- Put this into a text file

Sample MPI Job script

```
#!/bin/bash

#SBATCH --job-name=mpi_test_job
#SBATCH --partition=general
#SBATCH --output=%x.%j.out
#SBATCH --error=%x.%j.err
#SBATCH --account=PI_UCID
#SBATCH --qos=low
#SBATCH --time=00:10:00
#SBATCH --ntasks=256
#SBATCH --tasks-per-node=128
#SBATCH --mem-per-cpu=2G

# Run application commands
srun /apps/testjobs/bin/mpihello
```

- This runs an MPI job named “mpi_test_job”, with 256 processes total, spread over 2 nodes. Default setting is 1 core per process/task, so this also allocates 512Gb memory total. Wall time is 10 minutes.

Sample Single GPU Job script

```
#!/bin/bash

#SBATCH --job-name=test_gpu_job
#SBATCH --output=%x.%j.out
#SBATCH --error=%x.%j.err
#SBATCH --partition=gpu
#SBATCH --account= PI_UCID
#SBATCH --qos=low
#SBATCH --time=00:20:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --gres=mps:25

# Load application environment
module load CUDA

# Run application commands
nvidia-smi
```

- This runs a GPU job named “test_gpu_job”, with 8 cpus and a 25% weighted* time access to a single GPU. Wall time is 20 minutes.
- *Cycles of GPU based on total of all user percentages divided by requested percentage.

Sample Multi GPU Job script

```
#!/bin/bash

#SBATCH --job-name=test_gpu_job
#SBATCH --output=%x.%j.out
#SBATCH --error=%x.%j.err
#SBATCH --partition=gpu
#SBATCH --account=PI_UCID
#SBATCH --qos=low
#SBATCH --time=00:20:00
#SBATCH --ntasks=2
#SBATCH --cpus-per-task=32
#SBATCH --gres=gpu:2

# Load application environment
module load CUDA

# Run application commands
nvidia-smi
```

This runs a GPU job named “test_gpu_job”, with 64 cpus and full access to 2 GPUs. Wall time is 20 minutes.

Check the sample job scripts in `/apps/testjobs`

Manage Jobs - Overview

- Workload Manager
  - SLURM documentation:
    - "User Manual" on head node (accessible through Web Portal)
    - The Source: SLURM Documentation
  - man pages (sbatch, squeue, etc.)
  - Access methods

    SSH to head node
- Common job tasks

Submitting jobs	Pausing jobs
Submitting scripts	Resuming jobs
Running parallel jobs	Canceling jobs
Listing jobs	

Manage Jobs – Submit via CLI

Submit a job script

- $ **sbatch my_script**
- Submitted batch job 1234

Listing jobs

- For current user in Pending, Running, Suspended states:
  - **$ squeue –u xiss**

```
JOBID PARTITION   NAME    USER ST TIME  NODES NODELIST(REASON)
1234    lowpri uname.sh  xiss PD 0:00    2     (Priority)
```

For a more detailed query on active job:

- **$ scontrol show jobid=1234**

```
JobId=2 JobName=simple.job
UserId=xiss(1001) GroupId=xiss(1001) MCS_label=N/A
Priority=4294901759 Nice=0 Account=(null) QOS=normal
JobState=COMPLETED Reason=None Dependency=(null)
…
```

Canceling jobs

- $ **scancel 1234**

Show information about an active or completed job

- $ **slurm_jobid 1234**

Job States

CA CANCELLED - Job was explicitly cancelled by the user or system administrator. The job may or may not have been initiated.

CD COMPLETED - Job has terminated all processes on all nodes with an exit code of zero.

CF CONFIGURING - Job has been allocated resources but are waiting for them to become ready for use (e.g. booting).

CG COMPLETING - Job is in the process of completing. Some processes on some nodes may still be active.

F FAILED - Job terminated with non-zero exit code or other failure condition.

NF NODE_FAIL - Job terminated due to failure of one or more allocated nodes.

PD PENDING - Job is awaiting resource allocation.

R RUNNING - Job currently has an allocation.

RD RESV_DEL_HOLD - Job is held.

RH REQUEUE_HOLD - Held job is being requeued.

RQ REQUEUED - Completing job is being requeued.

ST STOPPED - Job has an allocation, but execution has been stopped with SIGSTOP signal. CPUS have been retained by this job.

S SUSPENDED - Job has an allocation, but execution has been suspended and CPUs have been released for other jobs.

TO TIMEOUT - Job terminated upon reaching its time limit.

Scheduling Policies and Limits

- Walltime limit
  - 72 hours
  - The jobs can be requeued at 72 hours interval
- SU charges
  - No SU charges on qos=high_PI, qos=low and qos=debug
  - 300,000 SU is allotted per PI group on qos=standard
  - qos=low is preemptable by qos=standard and qos=high_PI
  - If job submitted in qos=low is preempted, the jobs will be requeued once the resource becomes available.

Requeuing job

```
#!/bin/bash -l
#SBATCH --job-name=dam-break
#SBATCH --output=%x.%j.out
#SBATCH --error=%x.%j.err
#SBATCH --partition=general
#SBATCH --nodes=1
#SBATCH --open-mode=append
#SBATCH --ntasks-per-node=32
#SBATCH --qos=low
#SBATCH --mem-per-cpu=4G
#SBATCH --account=PI_ucid
#SBATCH --time=3-00:00:00
#SBATCH --requeue
#SBATCH --mail-type=ALL
#SBATCH --mail-user=ab1234@njit.edu

# Load the modules
module load foss/2022b OpenFOAM
source $FOAM_BASH

# Run the job using
requeue_job mpirun interFoam -parallel
```

Append the output to an exiting output file once requeued

Sample job script in /apps/testjobs/requeue

Memory Requirement

- By default, **mem-per-cpu=4G** is implemented unless specified in the job script
- Maximum **mem-per-cpu** is allowed up to **4G**
- If you need more memory, for single core job, increase **--ntasks-per-node**
- If **--mem** is used, then number of cores for the job will be calculated based on 4G per core for SU calculation.

Waiting for Your Job To Run

- Queue wait time depends on many factors
  - System load
  - Resources requested
    - nodes, cores, large memory, gpus, software licenses
    - **reduced priority for users or groups using a lot of resources**
- Check the running jobs in QoS
  - `squeue -q [QoS]`

Common inquiries

checkload	• sinfo but more details
checkq	• squeue but more details
slurm_jobid	• Show information about a running or queued job
sq	• Display pending job/queue info in a helpful way, You can also check the last job details with `sq`
quota_info	• Show space and SU quotas for self or others
listqos	• Show all QOSes or members of QOSes

Some Common Problems

After using sbatch, the job disappears in 30 seconds and there's no result output.

- Check the details with slurm_jobid [JOBID], use --err and --out
- Use sq if you are unsure about the job id.

Invalid account or account/partition combination specified.

- Check --account
- Use quota_info $LOGNAME

```
JOBID PARTITION   NAME   USER ST TIME  NODES NODELIST(REASON)
1234    general uname.sh  xiss PD 0:00   2 (ReqNodeNotAvail, Reserved for
maintenance)
```

- Jobs that do not end before the maintenance window begins will be held until the maintenance is complete

```
JOBID PARTITION   NAME   USER ST TIME  NODES NODELIST(REASON)
1234    general uname.sh  xiss PD 0:00      2 (MaxCpuPerAccount)
```

- listqos high_$PI
- squeue -q high_$PI

```
JOBID PARTITION   NAME   USER ST TIME  NODES NODELIST(REASON)
1234    general uname.sh  xiss PD 0:00      2 (AssocGrpBillingMinutes)
```

- Your PI group have reached the limit of SU in standard

Interactive Batch Jobs

Interactive, but handled through batch system	Resource limits same as standard batch limits
Useful for tasks forbidden on login nodes	Debug parallel programs Run a GUI program that's too large for login node Quickly test your code
May not be practical when system load is high	Long wait, same as standard batch job
To submit an interactive batch job (example)	***srun -p general -n 1 --ntasks-per-node=8 --qos=standard --account=PI_ucid --mem-per-cpu=2G --time=59:00 --pty bash***

Use Applications with GUI

Login to Wulver using

- ***ssh -X -Y ucid*@wulver.njit.edu**

Start an interactive session with X11 forwarding.

```
srun -p general -n 1 --ntasks-per-node=1 --qos=standard --account=PI_ucid --x11 --time=59:00 --pty bash
```

Load the modules

Launch the application

Wulver Maintenance

- Wulver will be temporarily out of service for maintenance once a month, specifically on the 2nd Tuesday, to perform updates, repairs, and upgrades.
- During the maintenance period, the logins will be disabled
- Jobs that do not end before the maintenance window begins will be held until the maintenance is complete

Resources to get your questions answered

Getting Started: Access to Wulver

List of Software: Wulver Software

HOW TOs: Conda Documentation

Installing Python packages via Conda

Request Software: HPC Software Installation

Contact: Please visit HPC Contact

Open a ticket: email to hpc@njit.edu

Consult with Research Computing Facilitator: HPC User Assistance

System updates

- Read Message of the Day on login
- Visit NJIT HPC News

NJIT

hpc@njit.edu

hpc.njit.edu

twitter.com/njit

facebook.com/NewJerseyInstituteofTechnology

linkedin.com/school/njit/