NJIT New Jersey Institute of Technology

High Performance Computing

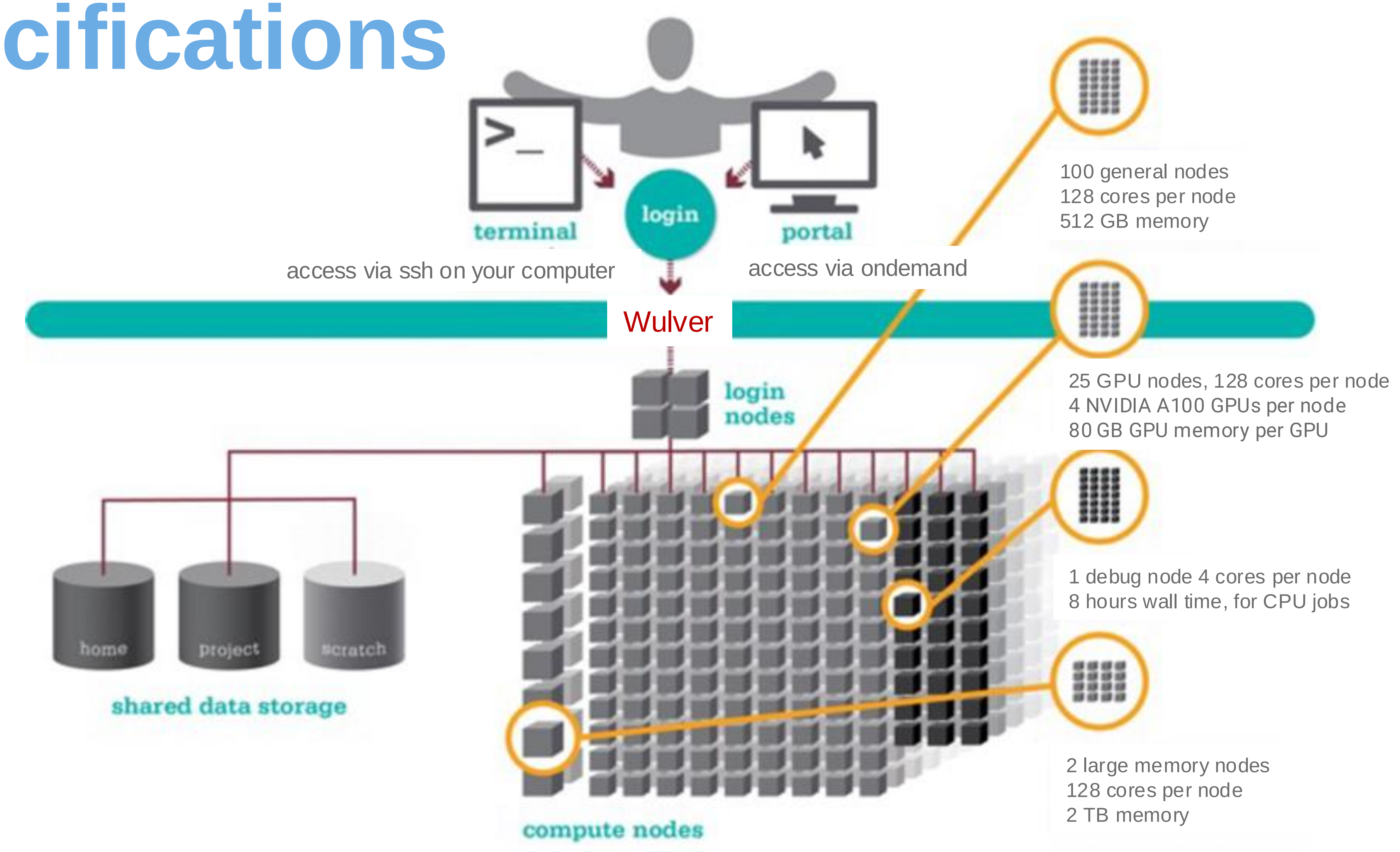
Introduction to Wulver: Accessing System & Running Jobs

Jan 29, 2025

Outline

- Wulver Specifications
- Access the software on Wulver
- Batch Processing
- Example of Slurm Jobs
- Manage Slurm Jobs
- Troubleshooting Common Issues
- Slurm Interactive Jobs and Use GUI Apps
- Contact Us

Wulver Cluster Specifications



access via ssh on your computer

access via ondemand

100 general nodes
128 cores per node
512 GB memory

25 GPU nodes, 128 cores per node
4 NVIDIA A100 GPUs per node
80 GB GPU memory per GPU

1 debug node 4 cores per node
8 hours wall time, for CPU jobs

2 large memory nodes
128 cores per node
2 TB memory

hpc.njit.edu/clusters/get_started_on_Wulver

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.

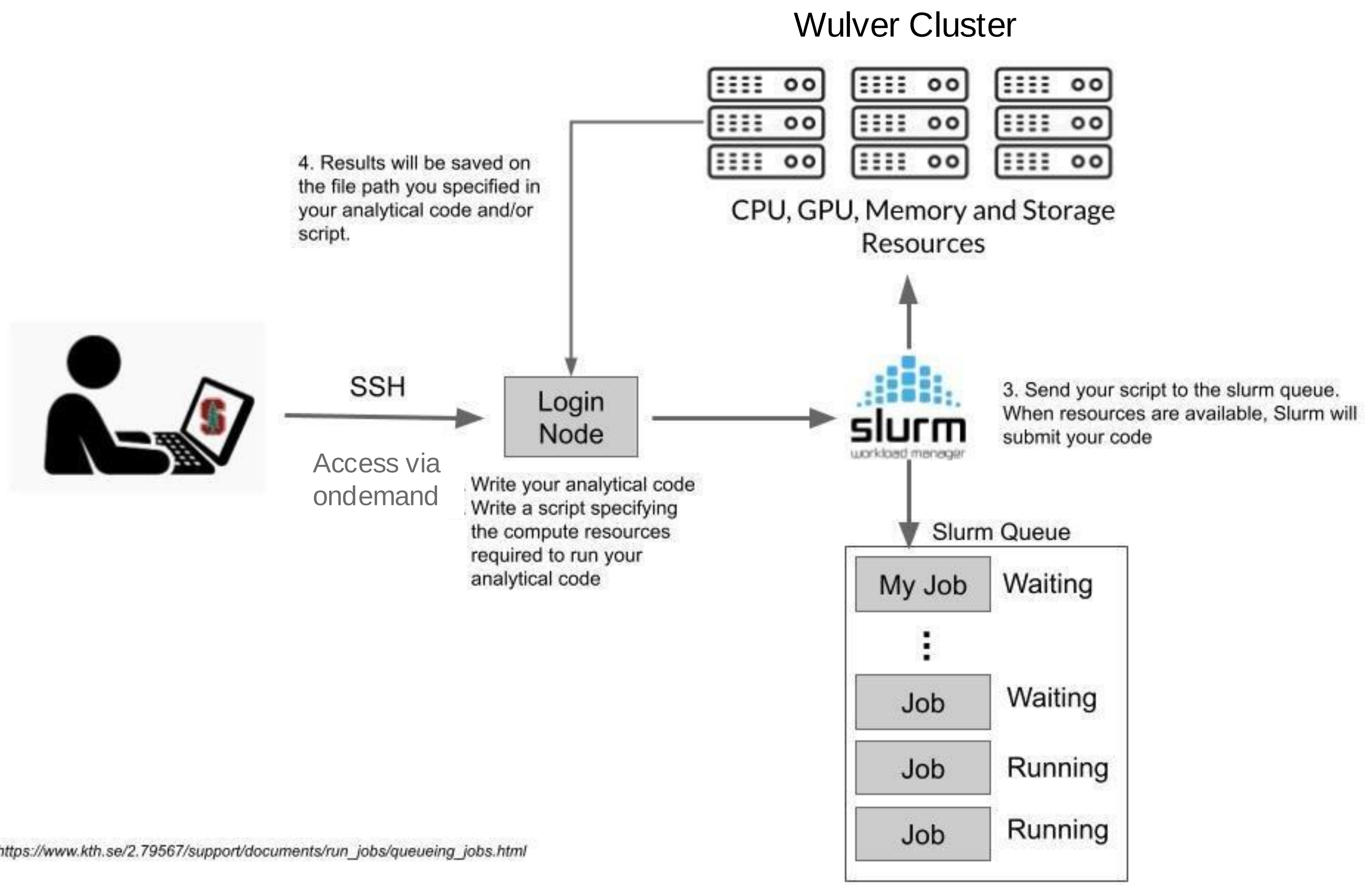
What modules do you have loaded?	**module list**
What modules are available?	**module spider** or **module avail**
Multiple versions of the same compiler	**module avail intel**
Add a software module to your environment	**module load CUDA**
Remove a software package from your environment	**module unload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

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

NJIT

Batch Processing

Why do supercomputers use queuing?


Wulver Cluster
4. Results will be saved on the file path you specified in your analytical code and/or script.
CPU, GPU, Memory and Storage Resources
SSH
Access via ondemand
Login Node
Write your analytical code
Write a script specifying the compute resources required to run your analytical code
slurm workload manager
3. Send your script to the slurm queue. When resources are available, Slurm will submit your code
Slurm Queue
My Job
Waiting
⋮
Job
Waiting
Job
Running
Job
Running

Source: https://www.kth.se/2.79567/support/documents/run_jobs/queueing_jobs.html

What is Slurm?

- Slurm is the predominant Open-Source scheduler for HPC compute
- Historically Slurm was an acronym standing for
  - **S**imple **L**inux **U**tility for **R**esource **M**anagement
- The Slurm scheduler provides three key functions:
  - it allocates access to resources (compute nodes) to users for some duration of time so they can perform work.
  - it provides a framework for starting, executing, and monitoring work (typically a parallel job such as MPI) on a set of allocated nodes.
  - it arbitrates contention for resources by managing a queue of pending jobs.

Manage Jobs – Options

Mandatory Options

Directive	Options	Description
--account=	account	PI’s UCID
--partition=	Partition	Request a partition of resources for job allocation (queue)
--time=	Time [[d-]h:]m[:s]	Maximum time limit on job allocation
--qos=	Job Priorities	Define the job priority

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

HPC Partitions

- Example of SU charges: (20 cores with 4 GPUs for 8 hours)
- SU = 20 x 8 x 3 = 480

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/

Job Submission Time Interval Formats

- Valid time formats (with a few exceptions) for `-t / --time=` option

Minutes	(`-t 10` is 10 minutes)
Minutes:Seconds	(…`10:30` is 10 minutes & 30 secs)
Hours:Minutes:Seconds	(`1:0:0` is 1 hr + 0mins + 0secs)
Days-Hours:Minutes:Seconds	(`7-1:10:30` is 7days + 1hr + 10mins + 30secs)
Days-Hours	(`7-0` is 7days + 0hrs i.e. 7 days)
Days-Hours:Minutes	(`7-4:10` is 7days + 4hrs + 10mins)

QoS

- 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
  - SUs will reset every year in mid-January with no carryover.
- 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
  - Use `listqos` command

Manage Jobs – Options

Additional Options

Directives	Options	Description
--ntasks=	Number of cpus	Number of CPUs (tasks) to be allocated
--nodes=	Node	Number of Nodes
--ntasks-per-node=	Numbers of cpus per node	Number of CPUs (tasks) per each node to be allocated
--mem=	Memory	Total memory of the job
--mem-per-cpu=	Memory per cpu	Memory to be allocated per each cpu
--gres=	Generic resources	Set the Number of gpus
--cpus-per-task=	Cpus per task	Number of CPUs per task
--requeue	Requeue	This is required when you want to continue the job after 72h walltime.

Directives	Options	Description
--error=	File	Define standard error file
--out=	File	Define standard output file
--input=	File	File used for standard input
--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).
--dependency=	Job dependency	Set the job dependency when submitting multiple jobs

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

General Application Workflow

- Log into cluster with ucid and password
- 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
  - Update the software modules
- Submit the job file using `sbatch`

Examples of Slurm Jobs

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, a wall time limit of 20 minutes.
- QOS is required. Account is recommended.
- `%x.%j` expands to JobName.JobID and prints this into a text file

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

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 --ntasks-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.

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

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 sample job scripts in `/apps/testjobs/workshop`

Sample 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=2
#SBATCH --gpus-per-task=2

# Load application environment
module load CUDA

# Run application commands
nvidia-smi
```

- This runs a GPU job named “test_gpu_job”, with 2 cpus and 2 GPUs

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

Manage Slurm Jobs

Manage Jobs - Overview

- SLURM documentation:
    - “User Manual” on head node (accessible through Web Portal)
    - The Source: SLURM Documentation
  - man pages (sbatch, squeue, etc.)

- Common job tasks
  - Submitting jobs
  - Running parallel jobs
  - Listing jobs
  - Resuming jobs
  - Canceling 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 $LOGNAME`

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

For a more detailed query on active job:

- `scontrol show jobid=1234`

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

Canceling jobs

- `scancel 1234` - Cancel job ID 1234
- `scancel --me` - Cancel all your jobs

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.**
- **RQ REQUEUED** - Completing job is being requeued.
- **PR PREEMPTED** - The job was terminated because of preemption by high priority job.
- 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.

sbatch Example: Memory

Submit a batch job

```
> sbatch --mem=1000 my_work.bash Submitted batch job 44003
```

Options used

--mem	Amount of reqeusted memory (K|M|G|T)

This Job will allocate 1000M of memory

SU Charges for Memory Allocation

```
> srun --partition=gpu --nodes=1 --ntasks-per-node=8 --mem=160G --
qos=standard --gres=gpu:1 --time=2:00:00 --pty bash
```

What will be SU Charge for this job?

slido

Please download and install the Slido app on all computers you use

What will be SU charge for this job?

```
srun --partition=gpu --nodes=1 --ntasks-per-node=8 --mem=160G --qos=standard --gres=gpu:1 --time=2:00:00 --pty bash
```

ⓘ Start presenting to display the poll results on this slide.

SU Charges for Memory Allocation

```
> srun --partition=gpu --nodes=1 --ntasks-per-node=8 --mem=160G --
qos=standard --gres=gpu:1 --time=2:00:00 --pty bash
```

- Number of CPUs - 8
- Memory Requested - 160G
- Based on 4G/core, number of equivalent cores - 160/4 = 40

SU Charges for Memory Allocation

```
> srun --partition=gpu --nodes=1 --ntasks-per-node=8 --mem=160G --
qos=standard --gres=gpu:1 --time=2:00:00 --pty bash
```

- Number of CPUs - 8
- Memory Requested - 160G
- Based on 4G/core, number of equivalent cores - 160/4 = 40
- SU Charges - 40 X 2 X 3 = 240

We will use whichever is the maximum

- Use `slurm_jobid` command to check SU usage

sbatch Example - Multiple apps in single script

Submit a batch job

```
> sbatch my_work2.bash
Submitted batch job 44005

> cat my_work2.bash
```

The ***srun*** commands utilize the resources in the allocation request.

The ***srun*** commands execute desired tasks **WITHIN** the set of requested resource.

They cannot use more/other than what was requested.

```
#!/bin/bash
#SBATCH --ntasks=128
#SBATCH --mem-per-cpu=4G
#SBATCH --time=60
...

srun --ntasks=100  app1  &
srun --ntasks=20   app2  &
srun --ntasks=8    app3  &
wait
bash my_cleanup_script.sh
...
```

Options used

`--ntasks`	Number of tasks
`--mem-per-cpu`	Memory required per CPU
`--time`	Wall time limit (minutes in our example)

Source: www.schedmd.com

sbatch Example : 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=standard
#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

sbatch Example - Job Dependencies

Submit sequence of three batch jobs

```
> sbatch --ntasks=1  --parsable pre_process.bash
45001
> sbatch --ntasks=128 --parsable --dependency=afterok:45001 do_work.bash
45002
> sbatch --ntasks=1  --parsable --dependency=afterok:45002 post_process.bash
45003
```

Options used

`--ntasks`	Number of tasks and by default the number of cores
`--dependency`	Job dependency

https://slurm.schedmd.com/sbatch.html#OPT_dependency

Job Dependency Options

after:job_id[:job_id...]
This job can begin execution after the specified jobs have begun execution.

afterany:job_id[:job_id...]
This job can begin execution after the specified jobs have terminated (regardless of state).

afterburstbuffer:job_id[:jobid...]
This job can begin execution after the specified jobs have terminated and any associated burst buffer stage out operations have completed.

afternotok:job_id[:job_id...]
This job can begin execution after the specified jobs have terminated in some failed state (non-zero exit code, node failure, timed out, etc).

afterok:job_id[:job_id...]
This job can begin execution after the specified jobs have successfully executed (ran to completion with an exit code of zero).

aftercorr:job_id
A task of this job array can begin execution after the corresponding task ID in the specified job has completed successfully.

singleton
This job can begin execution after any previously launched jobs sharing the same job name and user have terminated

Waiting for Your Job To Run

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

Troubleshooting Common Issues

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`
squeue --start	• Jobs will be listed in order expected start time • Times are only estimates and subject to change
quota_info	• Show storage 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`

Some Common Problems(contd)

```
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
- `scontrol update JobId=Job_ID QOS=low`

Error message: cggroup out of memory handler

- Check the memory requirement
- You probably need to increase memory on --mem or the number of cpus from --mem-per-cpu
- If nothing works, then the problem is likely due to incorrect setup of the problem.

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

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

Slurm Interactive Jobs and Use GUI Apps

Interactive Batch Jobs

Interactive, but handled through batch system

Resource limits same as standard batch limits
Use `srun` or `salloc` command

Useful for tasks forbidden on login nodes

Debug parallel programs

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 --nodes=1 --ntasks-per-node=8 --qos=standard --account=PI_ucid --mem-per-cpu=2G --time=59:00 --pty bash
```

Using Applications with GUI on OnDemand

Login to ondemand.njit.edu

Go to “Interactive Apps” and select the application from the list.

If you don’t find the app, select Linux Desktop

Once are you connected, select “Terminal Emulator” from “Applications” option from top left

Reminder

- 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
  - Reduce the walltime in the job script to run the job

- Date: Every Monday and Wednesday
- Time: 2:00–4:00 p.m.
- Location: GITC 2404
- Meet with our student consultants and ask any questions you have about using HPC resources.
- There's no need to create a ticket in advance.

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

Access to OnDemand Open OnDemand

Contact: Please visit HPC Contact

Open a ticket: email to hpc@njit.edu

Consult with Research Computing Facilitator: Facilitator Calendar Appointment

System updates

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

NJIT

hpc@njit.edu

hpc.njit.edu