Northwestern
INFORMATION TECHNOLOGY

# Intro to Quest

Northwestern IT Research Computing and Data Services

Andrew Block, Wenceslao Hernandez Velasquez, Julia Giannini

June 26, 2025

## Overview

## Technical Support

quest-help@northwestern.edu

# Requesting Help

- Request a consult or open a ticket!
  - quest-help@northwestern.edu
- Office hours:
  - Every Monday, 3-4 PM
  - Rooms 2202-2205, Mudd Library (2nd floor)
    - Across the outside hallway from the GIS Lab

## What is High Performance Computing (HPC)?

## High Performance Computing

HPC uses supercomputers and computer clusters to solve advanced computation problems. – Wikipedia

HPC encompasses solutions that are able to process data and execute calculations at a rate that far exceeds other computers. This aggregate computing power enables [scientists] to solve large problems that would otherwise be unapproachable. – HPE

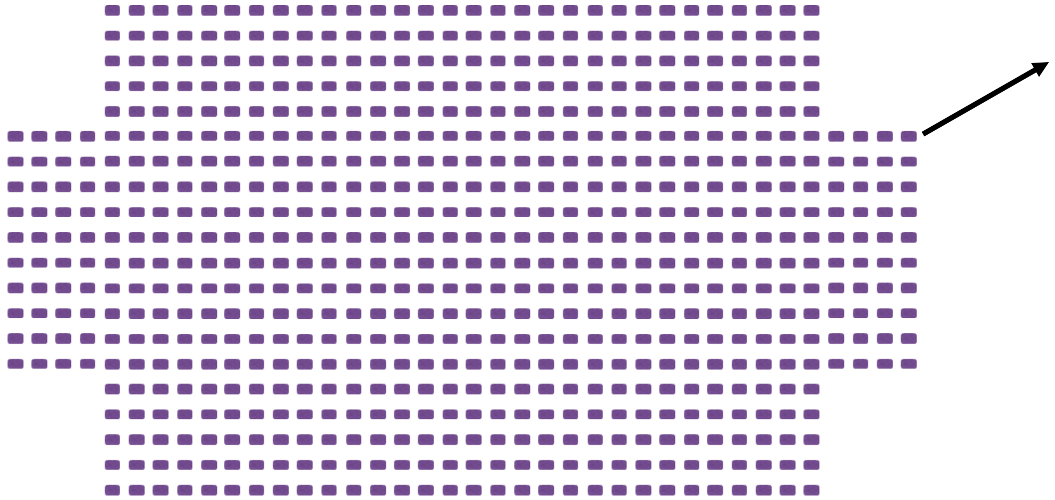
HPC is technology that uses clusters of powerful processors, working in parallel, to process massive multi-dimensional datasets (big data) and solve complex problems at extremely high speeds. - IBM

# Quest!

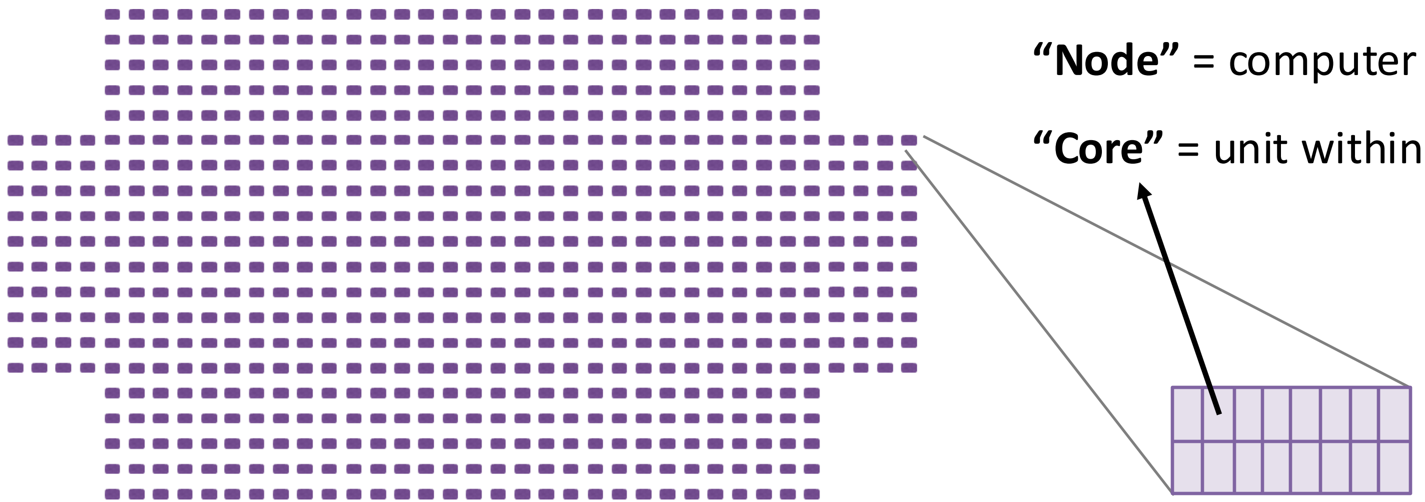
# HPC at Northwestern

- Host big databases and facilitate computing for astrophysics, biology, etc.
- Enable Genomics Research and Pipelines with GCC
- Be able to use the power of multiple computers
- Quest:
  - 5,500 active users, 1231 compute nodes, 80,390 CPU cores, 304 GPU Cards, 750 research software, 78 knowledge base articles, 28 training recordings

# Quest consists of ~1200 nodes



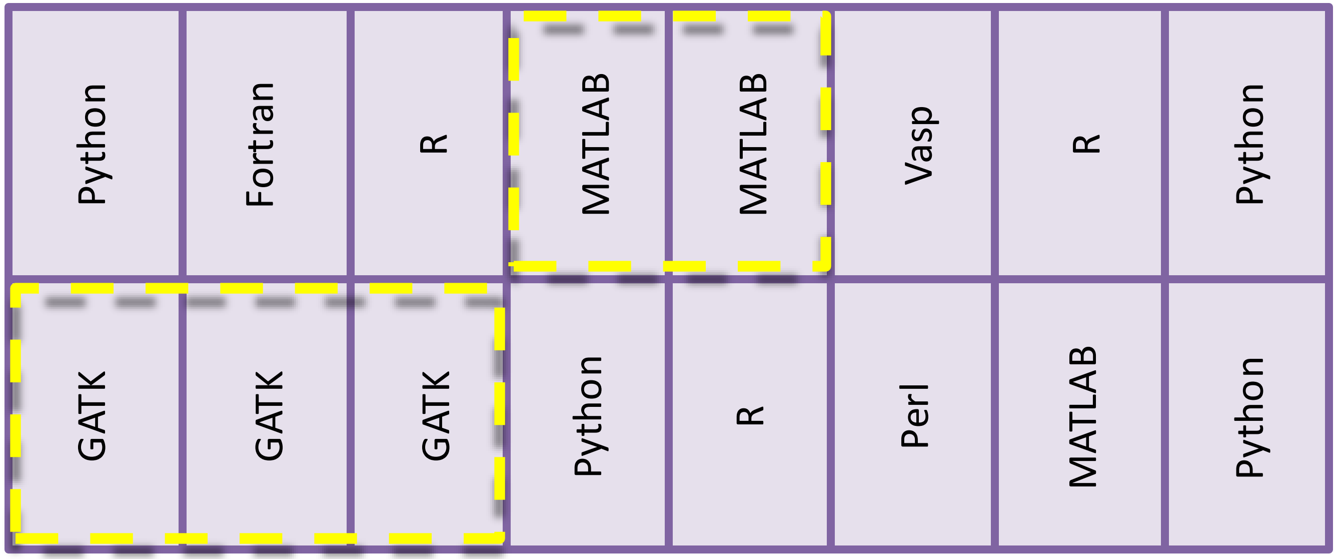
# Each node consists of 52-128 cores



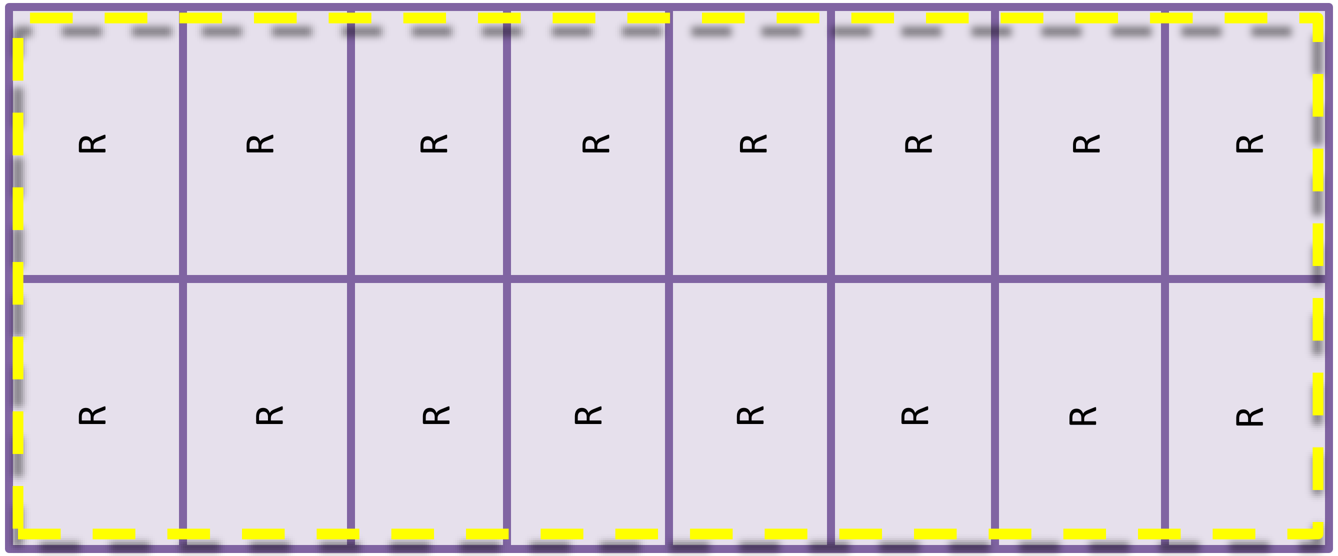
“**Node**” = computer

“**Core**” = unit within CPU/processor

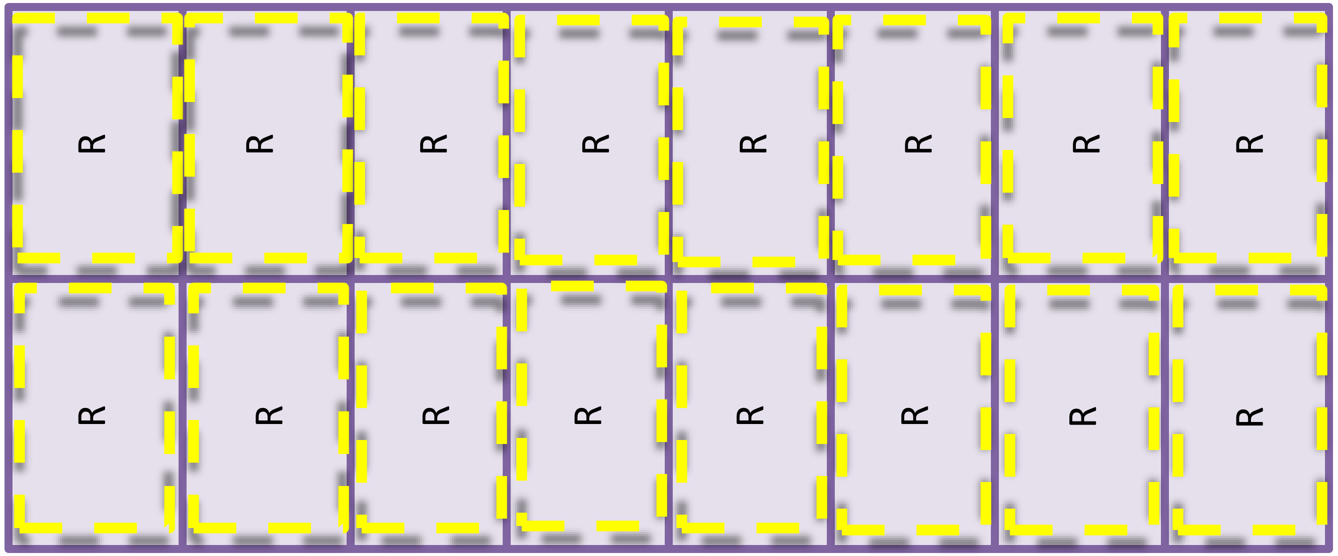
# Each node can run separate jobs on each core



# A node can run a job with 52-128 cores at a time

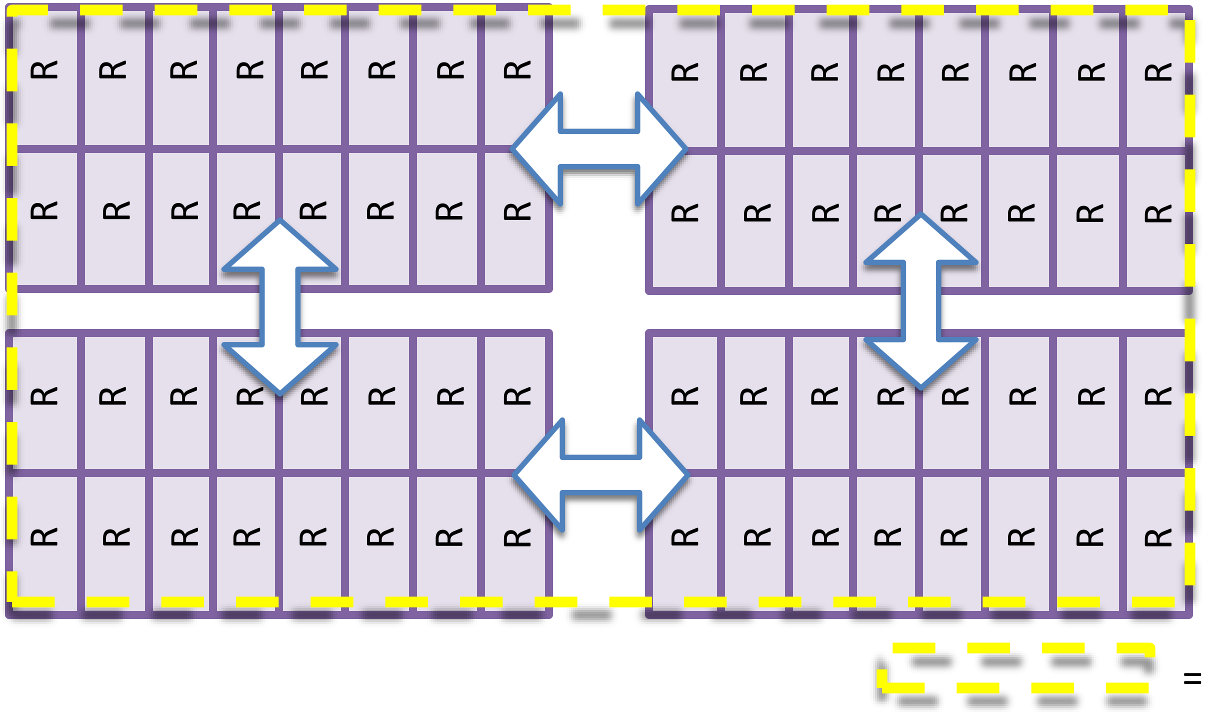


# You can run thousands of jobs at a time



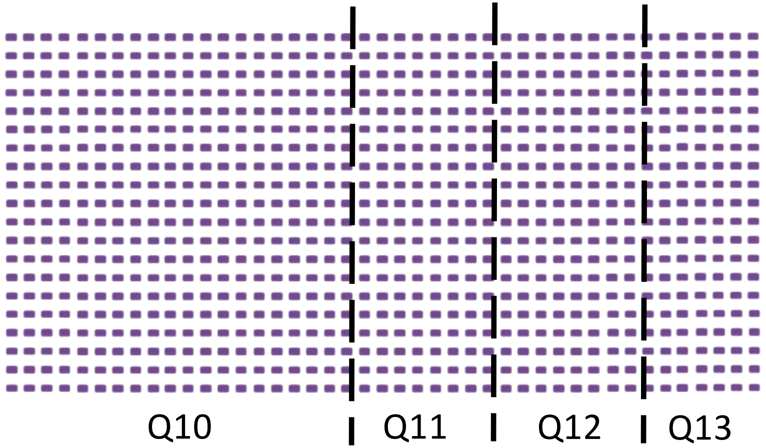
= a single job

# Jobs can even run across multiple nodes



# Technical specifications of nodes vary

There are several generations of nodes!

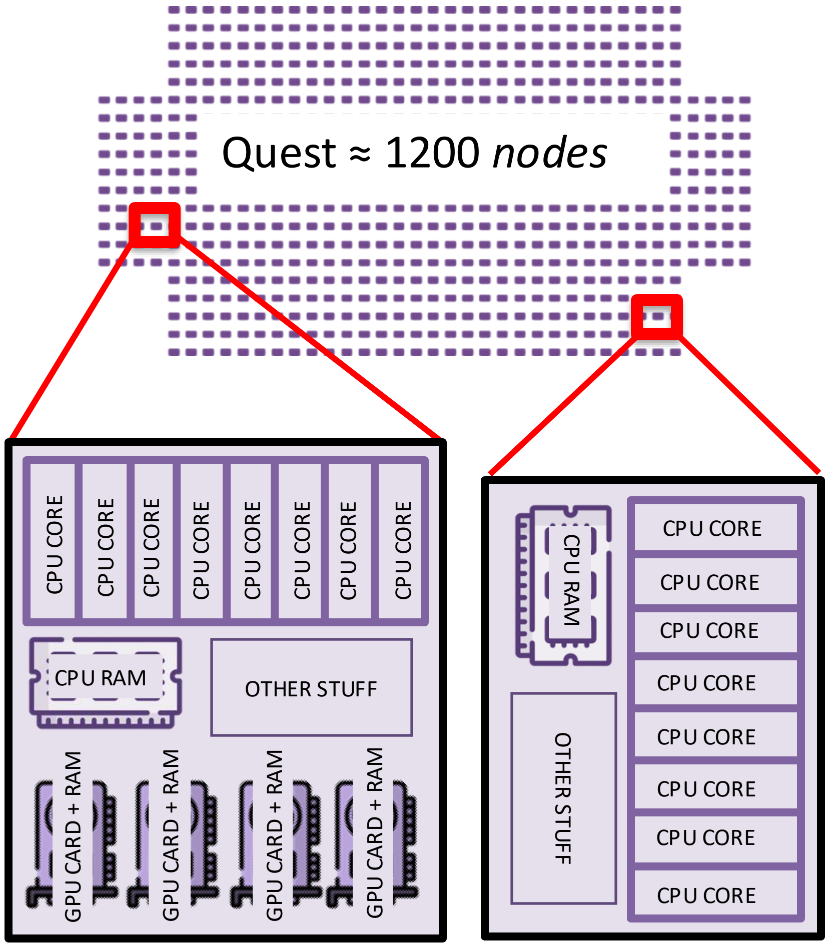


| Generation | # nodes | # cores per node | Total Schedulable memory (RAM) | Memory per core |
|---|---|---|---|---|
| Q10 | 555 | 52 | 166 GB | ~3.2 GB |
| Q11 | 209 | 64 | 221 GB | ~3.5 GB |
| Q12 | 214 | 64 | 221 GB | ~3.5 GB |
| Q13 | 140 | 128 | 473 GB | ~3.7GB |
| Your laptop (probably) | 1 | 10-25 | 16-128GB | ~2-8GB |

# Computing Resources

Each *node* has *schedulable resources*:

- 40-128 *CPU cores*
- 192-512 GB of *CPU RAM* (total)
- 0-4 *GPU cards*
- 40-80 GB of *GPU RAM* (per card)



# Specialty (general access) compute nodes on Quest

- High memory node available though General Access
  - 1.5 TB of RAM
  - 52 CPU cores

- GPU nodes available through General Access --> parallelization & speed
  - 58 nodes with 2 or 4 GPU cards each

## Quest technical details

- Cluster network interconnect: Infiniband
  - High speed communication between nodes, storage, etc.
- Storage: General Parallel File System (GPFS) ≈ 8PB (1PB =$10^{15}$B)
- Operating system: Red Hat Enterprise Linux (RHEL) 8.9
- Scheduler: SLURM (Simple Linux Utility or Resource Management

## Overview

## Allocations on Quest

- An allocation is a project group that allows you to do research and submit jobs on Quest
  - Provides:
    - Storage (personal and shared) space
    - Compute resources and the ability to submit jobs

- Types of allocations
  - Research I
  - Research II
  - Buy-in
  - Classroom

## How do I access files and folders on Quest?

- **File system is shared across all computers on Quest**

  → You can access your files & folders from anywhere

- **You will be working with 3 main folders:**

|  |  |
|---|---|
| ○ `/home/<netid>` | ← your personal home directory |
| ○ `/projects/<alloc-id>` | ← your project/allocation directories |
| ○ `/hpc/software` | ← where software modules live |

---

|  |  |
|---|---|
| ○ `/scratch /<netid>` | ← 5TB of temporary (30 day) storage |

# /home & /projects directories on Quest

| | `/home/<netid>` | `/projects/<allocation-id>` |
|---|---|---|
| Who has access | ▪ Just you | ▪ Everyone in the allocation |
| Storage | ▪ 80GB storage space | ▪ Much more space |
| Contains | ▪ Software, codes etc. | ▪ Data, outputs |
| **Check for space** | ▪ **“homedu”** | ▪ **“checkproject <alloc-id>”** |
| **Backups** | ▪ **Backed up every 24 hrs** | ▪ **NOT BACKED UP** |

## Who will have access to my files?

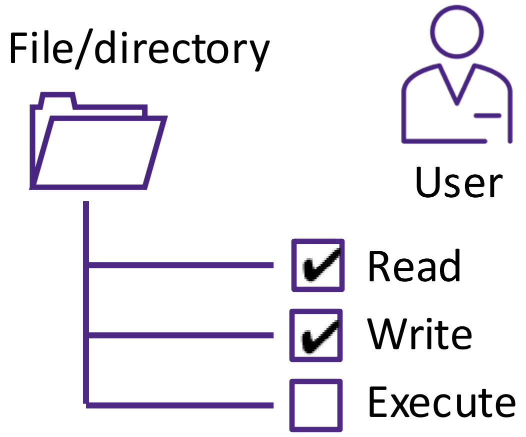
Files created in /projects

- Read/write permissions to all allocation members by default

Files created in /home

- Read/write permissions to only you by default

## What are “permissions”?



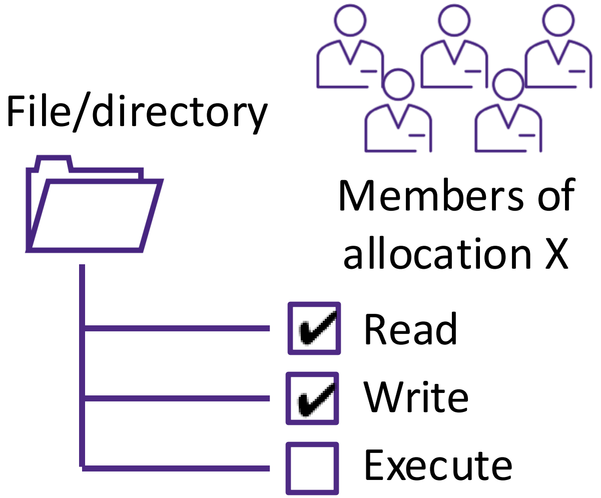
### Files you created:

- You will always have full read/write permissions.

### Files you did not create:

- You may not have read/write permissions.

# Group-level permissions



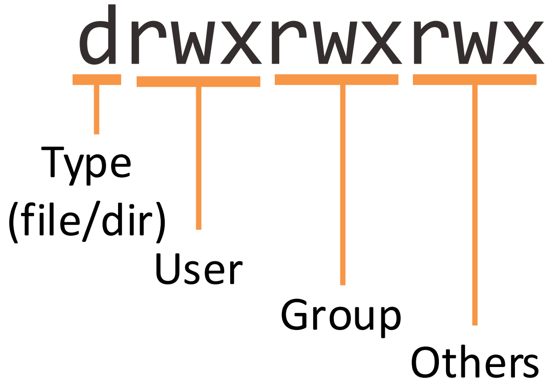
## Group permissions

- Permission are also defined at the “user group” level.
- To see which user groups you are a part of, run “groups”

```
[netid@quser21 ~]$ groups
netid     p30XXX     b10XX
```

## Permission strings

```
[abc1234@quser21 p12345]$ ls -l
total 2
drwxrwxr-x 1 abc1234 p12345 4096 Apr 15 11:00 codes
drwxrwxr-x 1 abc1234 p12345 4096 Apr 15 10:00 data
-rw-rw-r-- 1 abc1234 p12345  259 Apr 15 09:00 test.txt
```



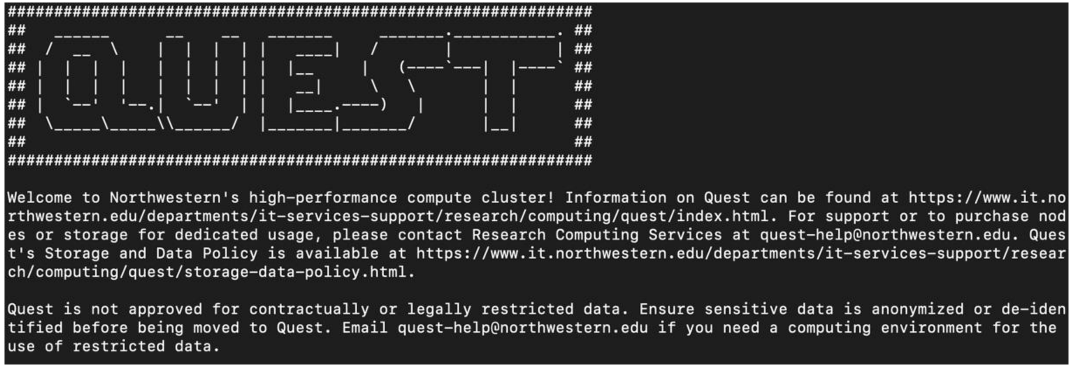
A bit more on permissions: [https://kb.northwestern.edu/70712](https://kb.northwestern.edu/70712)

# Overview

1. Introduction
2. Navigating Quest
3. <u>Ways to interact with Quest</u>
4. Data Transfers
5. How to submit jobs
6. Quest OnDemand

# Ways to interact with Quest

- Login nodes
  - For testing/development purposes can run things here
    - Limited to 4 CPUs and 4GB RAM
  - Submit **batch** or **interactive** jobs (terminal)
- Quest Analytics (web interface for interactive applications)
- Quest OnDemand (mainly **interactive** jobs)



```
###############################################################
##   ______      __    __   _______     _______.___________. ##
##  /  __  \    |  |  |  | |   ____|   /       |           | ##
## |  |  |  |   |  |  |  | |  |__     |   (----`---|  |----` ##
## |  |  |  |   |  |  |  | |   __|     \   \       |  |      ##
## |  `--'  '--.|  `--'  | |  |____.----)   |      |  |      ##
##  \_____\_____\\______/  |_______|_______/       |__|      ##
##                                                           ##
###############################################################

Welcome to Northwestern's high-performance compute cluster! Information on Quest can be found at https://www.it.no
rthwestern.edu/departments/it-services-support/research/computing/quest/index.html. For support or to purchase nod
es or storage for dedicated usage, please contact Research Computing Services at quest-help@northwestern.edu. Ques
t's Storage and Data Policy is available at https://www.it.northwestern.edu/departments/it-services-support/resear
ch/computing/quest/storage-data-policy.html.

Quest is not approved for contractually or legally restricted data. Ensure sensitive data is anonymized or de-iden
tified before being moved to Quest. Email quest-help@northwestern.edu if you need a computing environment for the
use of restricted data.
```

## Logging into Quest - Terminal

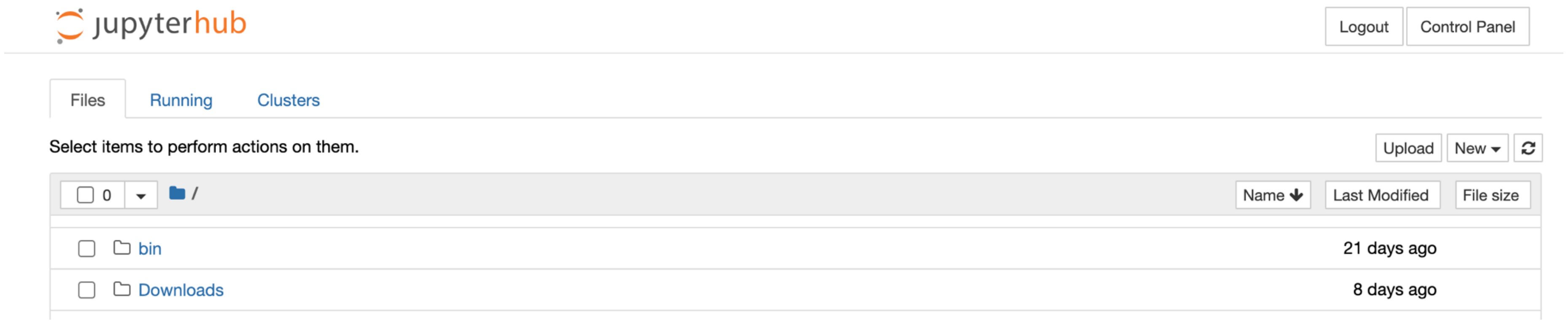
- Make sure you're connected to NU VPN
- Log in options
  - Terminal – SSH

```
ssh <net_id>@login.quest.northwestern.edu
```

    - Windows: use a terminal with Linux commands

# Ways to interact with Quest

- Analytics nodes (browser)
    - Run interactive Jupyter, R, or SAS sessions
    - Note usage guidelines on our website
        - Processes will automatically be killed if they overutilize CPUs or memory

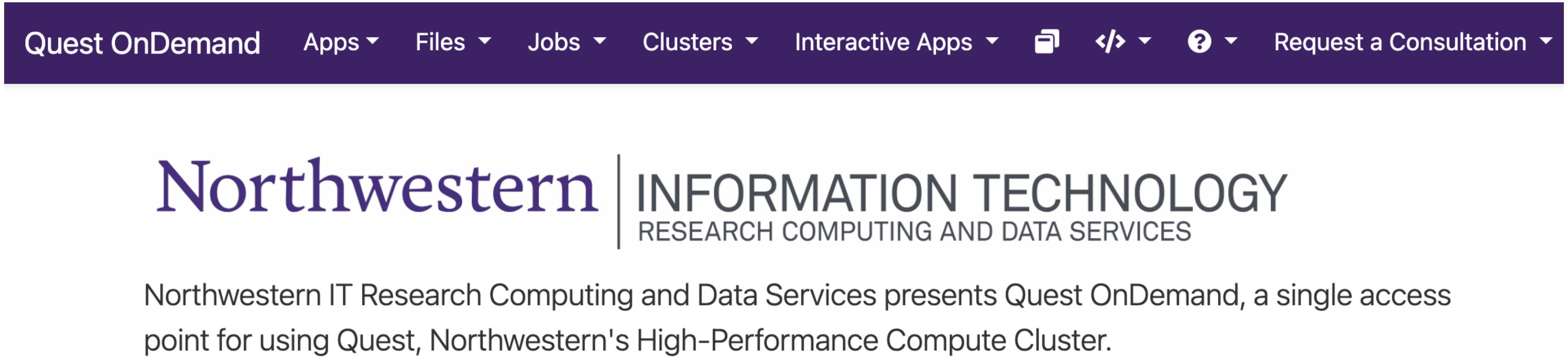


Select items to perform actions on them.

```
0  /
bin          21 days ago
Downloads     8 days ago
```

# Logging into Quest – Quest OnDemand

- Quest OnDemand (browser)
- Interactive jobs and GUIs
  - R, Juyper, VSCode, Matlab, Mathematica, Stata, and more!



Northwestern IT Research Computing and Data Services presents Quest OnDemand, a single access point for using Quest, Northwestern's High-Performance Compute Cluster.

# Overview

# Data Transfers

- Ways to transfer your data:
  - Globus
  - Quest OnDemand uploads
  - Command line utilities (`sftp`, `scp`, `rsync`, `wget`, `git`)

# Globus

- Globus completes data transfers between data storage locations and compute systems quickly and securely
  - Includes a web interface and a command line interface (CLI)
- Automated transfers
- Robust to connectivity problems
  - If you lose connection, Globus will continue where you left off
- Knowledge Base article

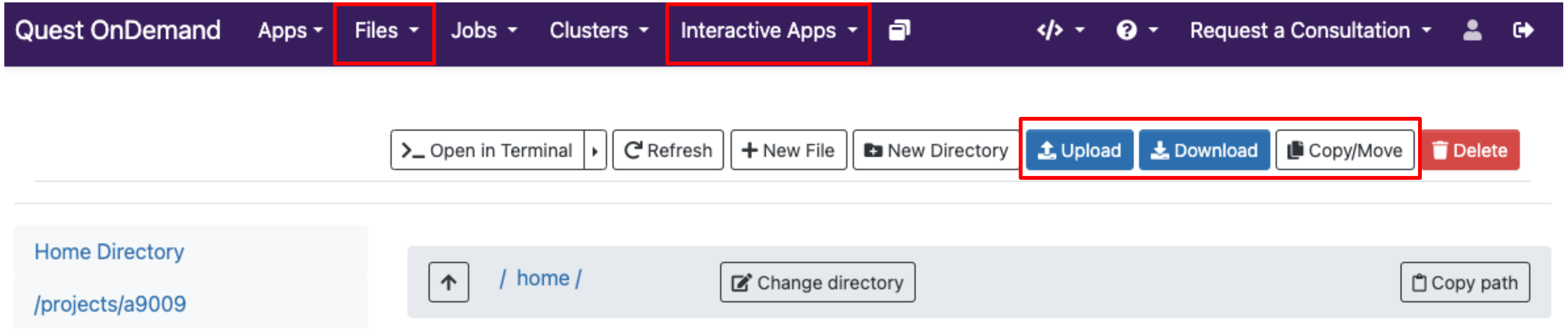
# Globus Endpoints

- Endpoint: A location where authorized users can transfer data to and from such as:
  - Quest
  - Your laptop
  - Your collaborator's laptops
  - Other university's HPCs

## Globus demo

# Quest OnDemand

- Web-based interface to run interactive applications on Quest compute nodes
- File explorer:
  - View, edit, and upload/download files
- More on Quest OnDemand later!



## File transfer through the command line

- Command line utilities: `sftp`, `scp`, `rsync`, `wget`, `git`

- SFTP – Secure File Transfer Protocol
  - Protects data from unauthorized access
  - `$ sftp <netid>@login.quest.northwestern.edu` – establish connection.
  - Upload files: `$ put <source_file> <destination_file>`
  - Downloading files: `$ get <source_file> <destination_file>`

## Overview

## Quiz!

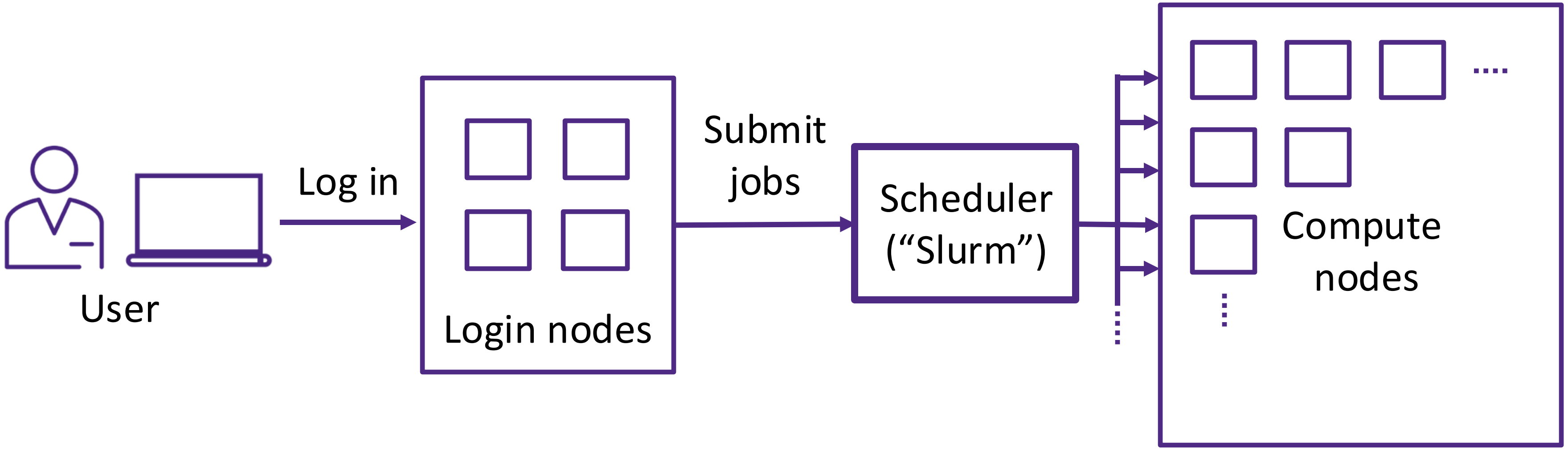
## Break

## Overview

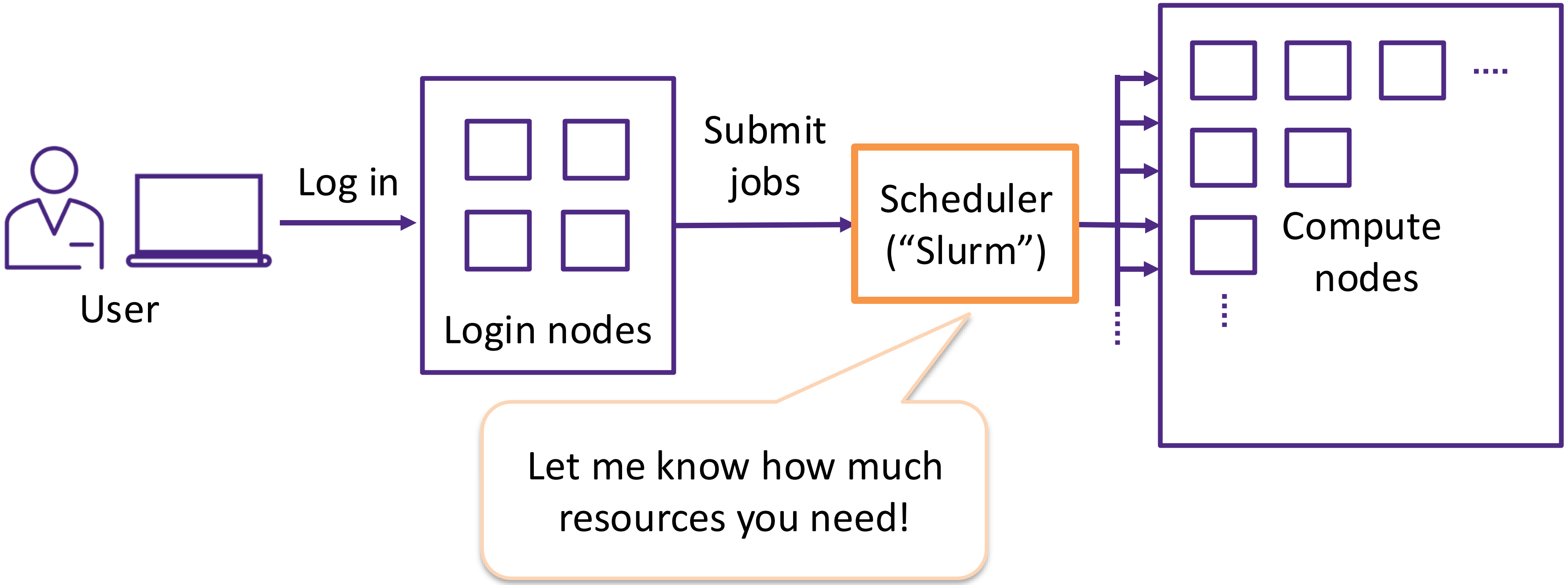
## Types of jobs: batch vs. interactive

- **Batch job**
  - Submit your job as a pre-written bash script
  - Benefit – submit & forget about it, high throughput
- **Interactive job**
  - Run interactive session on the *compute nodes*
  - Benefit – exploratory work, troubleshooting etc.

# Log into the login nodes & submit jobs to the scheduler



# Log into the login nodes & submit jobs to the scheduler



## What does a job script look like?

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog

## load modules (python)
module purge all
module load python


## run my program
python hello_world.py
```

Your script might look like this.
Let's break down the components!

## What does a job script look like?

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog

## load modules (python)
module purge all
module load python


## run my program
python hello_world.py
```

Tell Slurm what you want with these “headers”

Load any modules you need

Run your cool program!
(can be multiple commands)

## Side note: there are short-forms for headers

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog
```

=

```
#!/bin/bash
#SBATCH -A p12345
#SBATCH -p short
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH -t 00:10:00
#SBATCH --mem=3G
#SBATCH -J sample_job
#SBATCH --output=outlog
```

We will use the long-form in this workshop for clarity!

## Indicate allocation in “--account” or “-A”

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog

## load modules (python)
module purge all
module load python


## run my program
python hello_world.py
```

**Which allocation should I use?**

→ Try running “`groups <netid>`”

```
[abc1234@quser21 ~]$ groups abc1234
abc1234 p12345 b1000 e10001
```

## Indicate partition in “--partition” or “-p”

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog

## load modules (python)
module purge all
module load python


## run my program
python hello_world.py
```

**How do I know which partition to select?**

If your allocation starts with “**p**” or “**e**”:

→ “short” (4h) “normal” (48h) “long” (168 h)
→ “gengpu” for GPU
→ “genhimem” for high-memory

If your allocation starts with ”**b**”

→ generally, use allocation name (some special cases)
e.g. “**b1234**”

## Indicate number of nodes in “--nodes” or “-N”

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog

## load modules (python)
module purge all
module load python


## run my program
python hello_world.py
```

**How many nodes should I select?**

Unless your program explicitly states that it can run across nodes, default to nodes=1.

If your job doesn’t rely on software like **OpenMPI, MPICH, Intel-MPI,** then most likely nodes=1.

## Indicate number of cores in "--ntasks-per-node"

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog

## load modules (python)
module purge all
module load python


## run my program
python hello_world.py
```

**How many cores should I select?**

Unless your program is specifically designed to run on multiple cores, select "1".

Examples of multi-core:

Python's scikit-learn, R's DESEQ2 etc.

NOTE: you still need to explicitly tell the software!

## Indicate time in “--time” or “-t”

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog

## load modules (python)
module purge all
module load python


## run my program
python hello_world.py
```

**How much time should I request?**

→ Start with a conservative estimate.
→ Once you have a better idea of how much time your job will take, adjust accordingly.

## Indicate memory in “--mem” or “--mem-per-cpu”

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog

## load modules (python)
module purge all
module load python


## run my program
python hello_world.py
```

**How much memory should I request?**

Be careful as this affects your job’s pending time by a lot!

→ If you request a lot of memory, your job might be restricted to certain nodes.

→ You can take the same approach and start with a conservative estimate, but make sure to adjust later.

## Why not request maximum time and memory?

- **On Quest, job priority is determined using the user's <u>FairShare Score</u>**
    - FairShare Score makes sure that everyone gets their fair share
    - Score determined based on the <u>amount of resources you request</u>, not the amount that you end up using
    - Requesting a ton of resources will lower your job priority and increase pending time
    - HPC clusters consume a lot of energy
        - Strive for computational efficiency 🌱

## How to check resource utilization

Useful commands:

- `checkjob <job_id>` - detailed summary
- `seff <job_id>` - efficiency summary

```
[quest_demo@quser23]$ seff 2261088
---------------------------------------------
JOB EFFICIENCY
---------------------------------------------
Job ID: 2261088
Cluster: quest
User/Group: netid/netid
State: COMPLETED (exit code 0)
Cores: 4
CPU Utilized: 00:32:42
CPU Efficiency: 24.84% of 00:32:42 core-walltime
Job Wall-clock time: 00:32:42
Memory Utilized: 366.84 MB
Memory Efficiency: 3.66% of 10.00 GB
```

Can you spot any problematic trends that should be adjusted?

# How to check resource utilization

Useful commands:

- `checkjob <job_id>` - detailed summary
- `seff <job_id>` - efficiency summary

Can you spot any problematic trends that should be adjusted?

```
[quest_demo@quser43]$ seff 2261088
--------------------------------------------
JOB EFFICIENCY
--------------------------------------------
Job ID: 2261088
Cluster: quest
User/Group: netid/netid
State: COMPLETED (exit code 0)
Cores: 4
CPU Utilized: 00:32:42
CPU Efficiency: 24.84% of 00:32:42 core-walltime
Job Wall-clock time: 00:32:42
Memory Utilized: 366.84 MB
Memory Efficiency: 3.66% of 10.00 GB
```

# Set job name for ease of identification

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog

## load modules (python)
module purge all
module load python


## run my program
python hello_world.py
```

**How to decide job name?**

This is for your own convenience – Slurm will not care what you name your job. Something short and descriptive can be useful later.

# Set output file path

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog

## load modules (python)
module purge all
module load python


## run my program
python hello_world.py
```

**How to decide output file path?**

- Slurm expects you to specify a **full path** with a file name, not just a directory.
- Direct your output to **/projects**, not /home! Some programs write a lot of output and can overflow your /home

## Load software into the environment

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog

## load modules (python)
module purge all
module load python


## run my program
python hello_world.py
```

Load any modules you need

# Accessing/managing software on Quest

- Module system
  - Search for a module: `module spider <name_of_software>`

- **Virtual environments**
  - Anaconda virtual environments on Quest
- Containers
- Compile/install code from source

- Request a software installation

```
[abc1234@quser31 ~]$ module spider python
-----------------------------
python:
-----------------------------
    Versions:
      python/2.7.5
      python/2.7.18-gcc
      python/3.6.10
      python/3.8.12-gcc
      python/3.8.12-intel
      python/3.10.1
      . . .
```

# Execute commands/scripts

```
#!/bin/bash
#SBATCH --account=p12345
#SBATCH --partition=short
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:10:00
#SBATCH --mem=3G
#SBATCH --job-name=sample_job
#SBATCH --output=outlog

## load modules (python)
module purge all
module load python


## run my program
python hello_world.py
```

Run your cool program!
(can be multiple commands)

## Submit a batch job with sbatch

- `sbatch <name_of_submit_script>`

```
[abc1234@quser31 ~]$ sbatch submit_script.sh

Submitted batch job 7342820
```

## Types of jobs: batch vs. interactive

- **Batch job**
  - Submit your job as a pre-written bash script
  - Benefit – submit & forget about it
- **Interactive job**
  - Run interactive session on the compute nodes
  - Benefit – exploratory work, troubleshooting etc.

## Interactive job resources are requested similarly

```
[netid@quser23]$ srun --account=p12345 --time=1:00:00 -n 1 -p short --mem=1G --pty bash -l
----------------------------------------
srun job start: Wed Oct  5 14:11:43 CDT 2022
Job ID: XXXXXXX
Username: netid
Queue: short
Account: p12345
----------------------------------------
The following variables are not guaranteed to be the same in prologue and the job run
script
----------------------------------------
PATH (in prologue) :
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/usr/lpp/mmfs/bin:/opt/ibutils/bin:/opt/z
eek/bin:/home/netid/.local/bin:/home/netid/bin
WORKDIR is: /home/netid
----------------------------------------
[netid@qnode8075]$ # code away!
```

## “salloc” similar to “srun” but requires SSH

```
[quest_demo@quser23]$ salloc --x11 --account=p12345 --partition=short --nodes=1 --tasks-per-
node=1 --time=00:20:00
salloc: Pending job allocation 8575069
salloc: job 8575069 queued and waiting for resources
salloc: job 8575069 has been allocated resources
salloc: Granted job allocation 8575069
salloc: Waiting for resource configuration
salloc: Nodes qnode6702 are ready for job
[quest_demo@quser23]$ ssh -X qnode6702
Last login: Mon Dec 13 12:46:35 2021 from quser23
[quest_demo@qnode6702 ~]$ # Code away!
```

# Useful commands 1/2: see pending and running jobs

```
[netid@quser23]$ squeue -u netid # inspect pending or running jobs
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           2632440    p12345 sample_j tempuser  R       0:02      1 qnode5057
           2632441    p12345 sample_j tempuser  R       0:02      1 qnode5057
```

## Useful commands 2/2: see all jobs within a time window

```
[netid@quser23]$ sacct -X -u netid --starttime=9/24/22 --endtime=9/28/22 # past jobs
JobID               JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
1835975             testjob      short     p12345          1    TIMEOUT      0:0
2261088             testjob     normal     p12345          1  COMPLETED      0:0
2261164             testjob     normal     p12345          1 CANCELLED+      0:0
```

## Documentation and example jobs

**[Everything You Need to Know About Using Slurm on Quest]**

https://github.com/nuitrcs/examplejobs

## Exercise 1

# Exercise

1. Using the right git command, pull down the repository from https://github.com/nuitrcs/REU_Activities to your computer

2. Use your text editor of choice to open `start_example_submit.sh`

3. Finish the submission script!

Work with a friend if you like!

# Exercise - Answer

```
#!/bin/bash
#SBATCH --account=e32894  ## YOUR ACCOUNT pXXXX or bXXXX
#SBATCH --partition=short  ### PARTITION (buyin, short, normal, etc)
#SBATCH --nodes=1 ## how many computers do you need
#SBATCH --ntasks-per-node=1 ## how many cpus or processors do you need on each computer
#SBATCH --time=00:10:00 ## how long does this need to run (remember different partitions have restrictions on this param)
#SBATCH --mem-per-cpu=1G ## how much RAM do you need per CPU (this effects your FairShare score so be careful to not ask for more than you need))
#SBATCH --job-name=sample_job  ## When you run squeue -u NETID this is how you can identify the job
#SBATCH --output=outlog ## standard out and standard error goes to this file
#SBATCH --mail-type=ALL ## you can receive e-mail alerts from SLURM when your job begins and when your job finishes (completed, failed, etc)
#SBATCH --mail-user=email@u.northwestern.edu ## your email

module purge all
module load python/3.12.10

python --version
python slurm_test.py
```

## Overview

## Quest OnDemand Demo

## Exercise 2

# Thank You!

Questions?

quest-help@northwestern.edu

Request a Consultation

https://www.it.northwestern.edu/departments/it-services-support/research/computing/quest/