Computational Skills for Researchers
Intro to Quest

Janna Nugent, Sr. Bioinformatics Specialist
Research Computing Services

https://github.com/nuitrcs/intro_quest_workshop

Northwestern
INFORMATION TECHNOLOGY

Research Computing: the Consultants









- Astronomy
- Materials Science
- Data Science Support
- Bioinformatics
- Cloud Computing
- Data Workflow
- Visualizations
- Animations

quest-help@northwestern.edu

Research Computing: the Consultants

We are a team under Northwestern Information Technology with research backgrounds in computer science, astrophysics, materials science, data science, bioinformatics, and modeling and simulation.









quest-help@northwestern.edu

Research Computing: the Consultants









Providing infrastructure, training, and support for:

Computing

Data processing and analysis

Bioinformatics pipelines

Data management, sharing, and storage

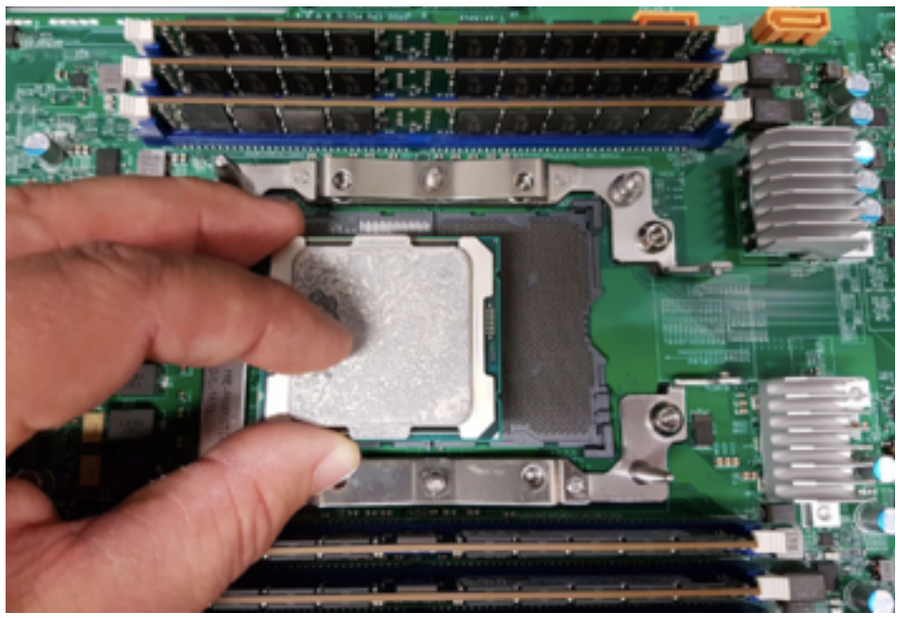
Software

quest-help@northwestern.edu

Quest: High Performance Computing Cluster

“node”: a computer

800



“core”: a processor

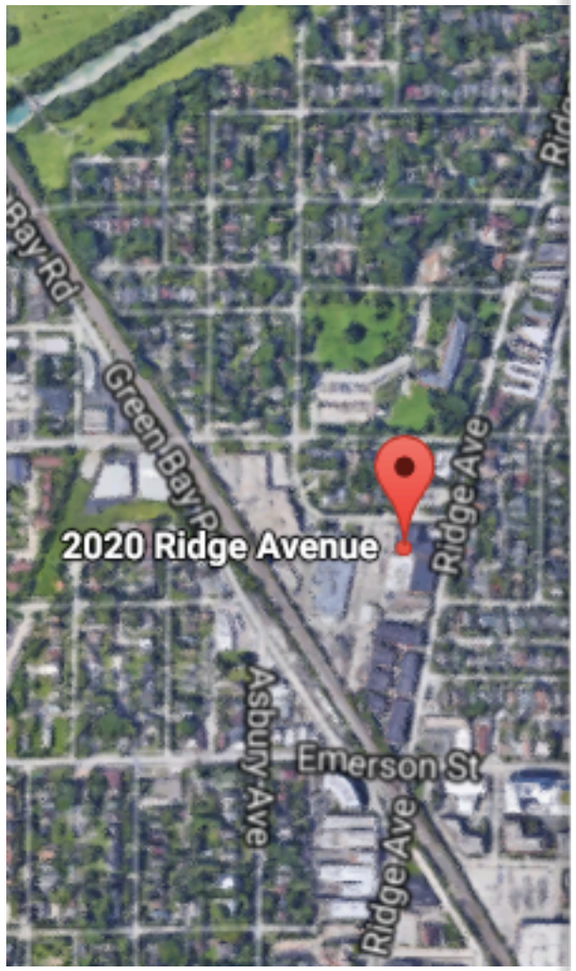
19,200

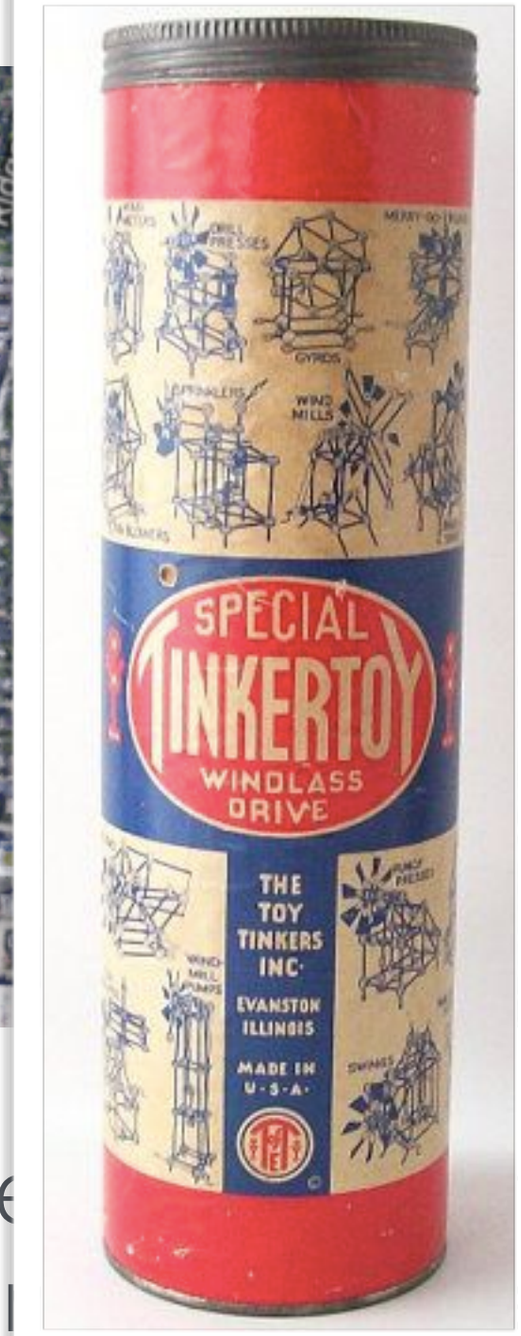
Quest: High Performance Computing Cluster

“infiniband”: high-speed inter-connect

nodes in racks

Quest: High Performance Computing Cluster





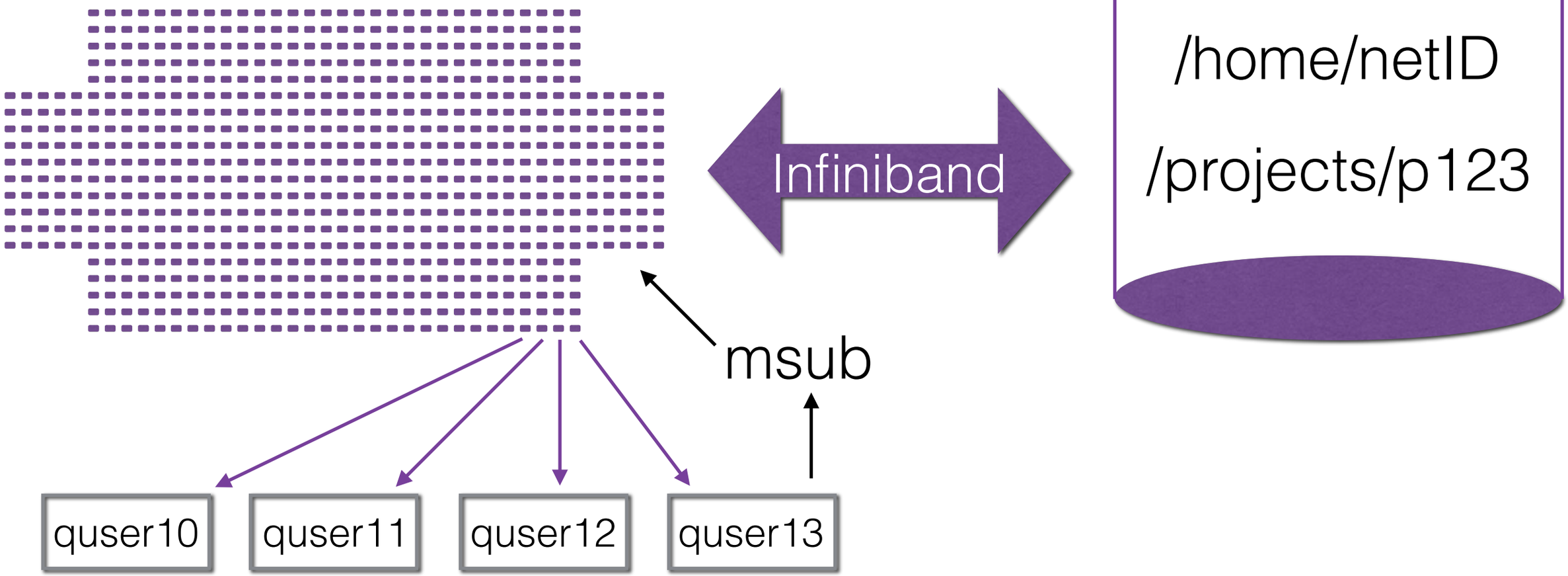
24 hour on-site se or authentication,
firewalls and i ction systems,
dedicated power station, high-throughput network

Quest: High Performance Compute Cluster

Allocation: compute hours, storage space, group

800 compute nodes

128 GB RAM



login nodes

ssh netID@quest.northwestern.edu

Analytics Nodes

The Quest Analytics Nodes allow users to run RStudio and SAS Studio in their web browser, backed by Quest file systems and nodes with more computational resources than available on a personal computer. They are available to all Quest users with an active allocation.

Backing up your data

Northwestern Box: unlimited, free, 15GB single file limit

RDSS & FSMRESfiles

Amazon AWS

Quest: High Performance Computing

ACTIVITY

Parallel Computing

Multiple jobs working independently

"embarrassingly parallel"

"pleasingly parallel"

high-throughput computing

Single job communicating across nodes

message passing:

MPI, OpenMP

Software

https://kb.northwestern.edu/quest-software

Quest: Logging in and Getting started

Mac users: launch the Terminal App:
ssh <netID>@quest.it.northwestern.edu

PC users: launch PuTTY (preferred) or FastX:
Hostname : quest.it.northwestern.edu
Username : your Northwestern NetID
Password : your Northwestern NetID password

Quest: Logging in and Getting started

1) login to Quest
   `$ssh` quest.northwestern.edu

2) where are you?
   `$pwd` -> this is your home directory

3) what is the name of your project allocation?
   `$groups` (your project is in **`/projects/<allocation>`**)

4) what is the status of your project?
   `$checkproject <allocation>`

5) how much space is used in your home dir?
   `$homedu`

Quest: Getting started with Cyberduck

To connect to Quest, start Cyberduck and then :

1) **Click Open Connection** in the upper left of the Cyberduck window
At the top of the Open Connection window that appears, Select SFTP (SSH File Transfer Protocol) from the drop-down menu.

2) **Enter quest.it.northwestern.edu** for server specification

3) **Enter your NetID** in the Username: box and leave the Password: box empty to prevent your NetID password from being saved in a file on your personal computer. Public Key Authentication is not supported.

4) **Click Connect**. You will see a Login failed window.

5) **Enter your NetID password** in the Password: field.

6) **Click Login.**

Quest: transfer intro.tar with Cyberduck

<u>On your local machine</u>

locate your intro.tar file in your Downloads directory

Drag intro.tar into your home directory in Cyberduck

<u>On the command line window that's logged into Quest</u>

$ ls confirm intro.tar is in the directory that you are in

$ tar xvf intro.tar to unpack the tar file

$ ls look for the unpacked files "submit_generic.sh" and "helloworld.py"

$ cat submit_generic.sh take a look at the header of this file

Quest: submit_generic.sh

```
#!/bin/bash
#MSUB -A <allocationID> ## <-- EDIT THIS TO BE YOUR ALLOCATION
#MSUB -q <queue_type>   ## <-- EDIT THIS TO BE YOUR QUEUE NAME
#MSUB -l nodes=1:ppn=1
#MSUB -l walltime=00:10:00
#MSUB -N sample_job
#MSUB -o outlog
#MSUB -e errlog

cd $PBS_O_WORKDIR       ## the directory from which the job was submitted
module load python      ## Load necessary modules (software pr libraries)

python helloworld.py    ## Run the program
```

Quest: submitting a job

login node

"msub submit.sh"

scheduler

```
#MSUB -A <allocationID>
#MSUB -q <queue_type>
#MSUB -l nodes=1:ppn=1
#MSUB -l walltime=00:10:00
```

compute node

compute node

compute node

compute node

compute node

Quest: sample bash submission script

What is the scheduler looking for in your script?

Behold, this is a BASH script: `#!/bin/bash`

Account: `#MSUB -A p20XXX`

Queue: `#MSUB -q short`

Number of nodes & cores: `#MSUB -l nodes=1:ppn=6`

Length of the job: `#MSUB -l walltime=04:00:00`

Name the job: `#MSUB -N projectname_mysoftware`

Generate an output log: `#MSUB -o outlog`

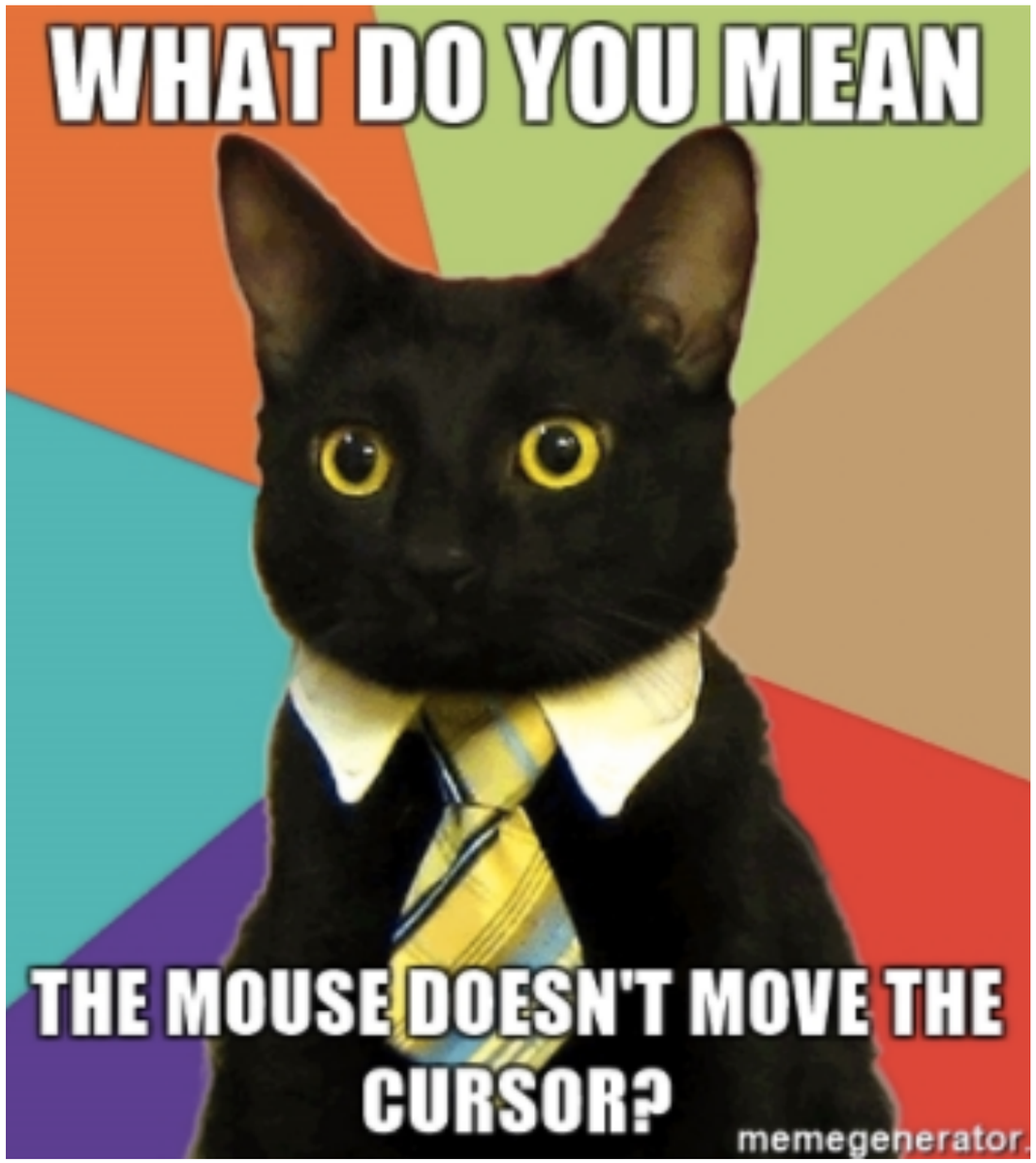
Generate an error log: `#MSUB -e errlog`

text editor: nano

- **$ nano submit_generic.sh**
  this will open the named file for editing or create it if it doesn't exist

- **type your text**: the mouse will not move the cursor - navigate with the arrow keys

- **save and quit**: commands are on the bottom of the screen - the "^" stands for "control"



Quest: sample bash submission script

1) submit your job

```
$msub submit_generic.sh
```

2) copy the job_id msub returns

3) where is your job in the queue?

```
$showq -u <your netID>
```

4) what is the status of your job?

```
$checkjob <job_id>
```

Jobs on Quest

https://kb.northwestern.edu/page.php?id=69247

Example Jobs

https://kb.northwestern.edu/page.php?id=70719

Questions?

email: quest-help@northwestern.edu

Research Computing Services

Northwestern
INFORMATION TECHNOLOGY