CLUSTERS 101

Guangzhen Jin

Senior Computational Scientist

PURDUE UNIVERSITY | Rosen Center for Advanced Computing

Outlines

- Introduction to Clusters
- Connecting
- File Systems
- Modules
- Jobs, Queues and SLURM
- Good Citizenship

Introduction to Clusters

Purdue Community Clusters

HPC (Brown, Bell, Negishi):
Multiple cores or nodes, probably MPI. Benefit from high-performance network and parallel filesystem. The vast majority of campus - 80% of all work!

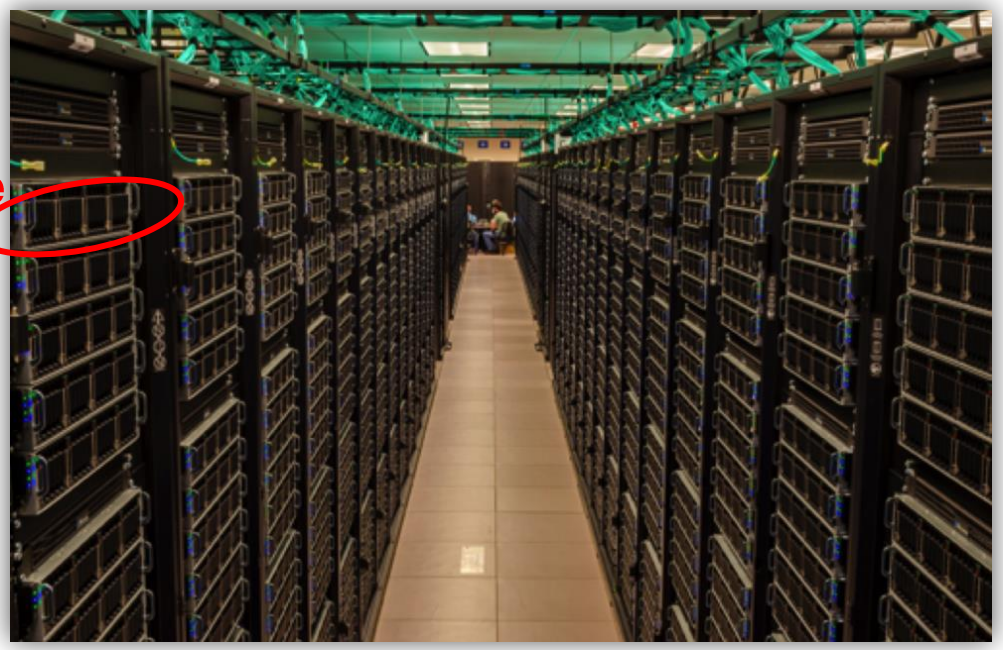
GPU Accelerated (Gilbreth):
Utilizes Nvidia V100, A10, A30, A100 GPUs for acceleration. Useful for Machine Learning, AI, Computational Chemistry, etc.

Scholar: Special case for teaching. Mostly MPI at first glance, but also highly tweaked for interactive use (tasks on front-ends, Jupyter notebooks, Rstudio, etc). Also couple GPUs and mini-Hadoop.

Anvil (Non-Community): Funded by National Science Foundation, enabling important discoveries across many different areas of science and engineering. Proposal required.

What is a Cluster?

- Hardware (compute nodes + interconnect + storage)
- Software (OS + compilers + libraries + apps + queue manager)
- Infrastructure (front-ends + power + cooling + data center + staff)



Node and Core on Clusters

- A **NODE** on a cluster is a single computing unit. Each node typically consists of processor(s), memory, storage, and network connectivity, and can communicate with other nodes in the cluster to exchange data and coordinate their work.
- A **CORE** is an individual compute unit ("slot") on the chip.

On the right:

You see:

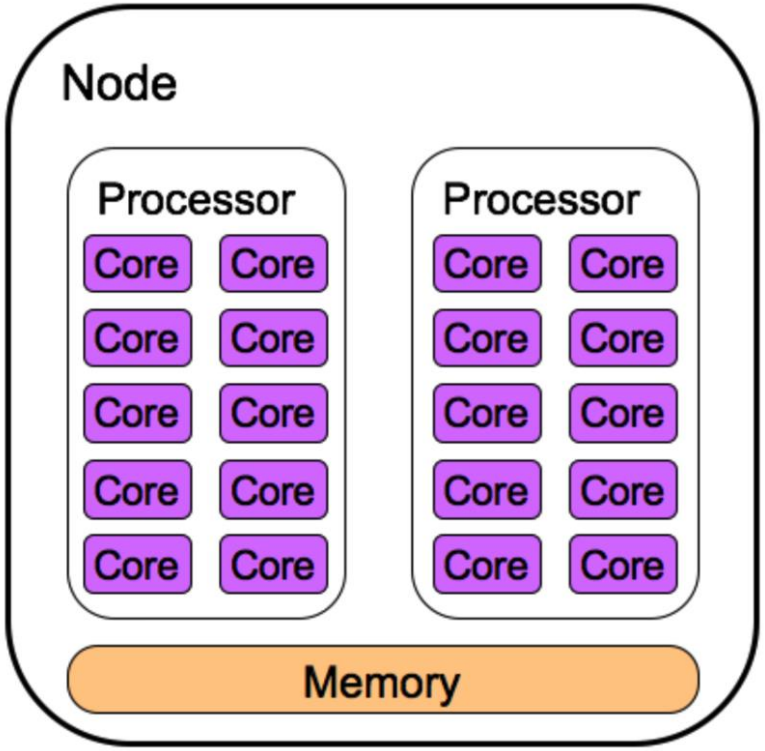
- 2 physical processors, 10 cores ea.

Queuing system sees:

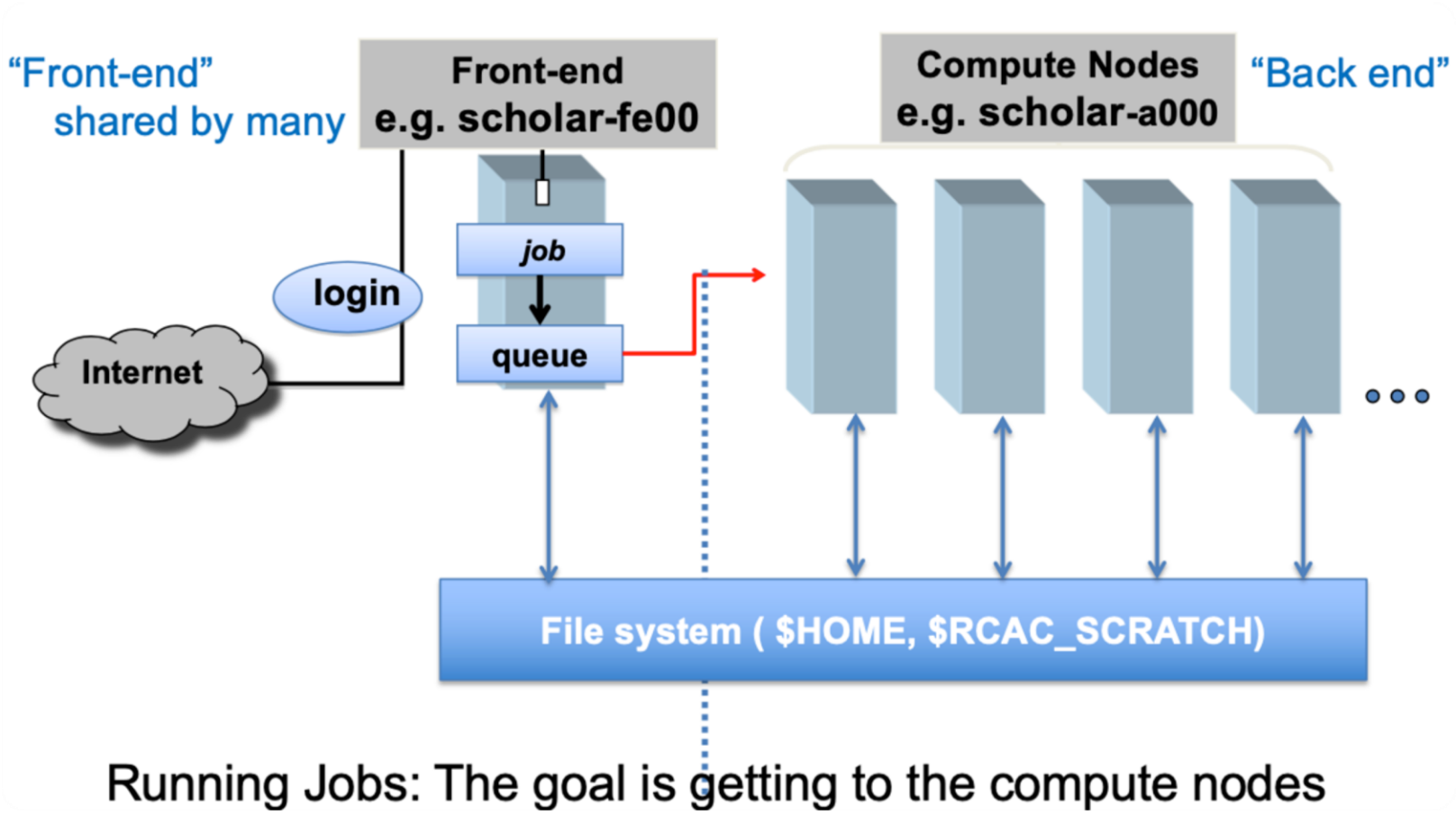
- 20 logical processors

From now on, we will be mostly concerned with cores (logical processors), not physical chips

(I ran a job on "5 CPUs" == "5 processors"== "5 cores")



Front-end vs Compute Node



Connecting - SSH

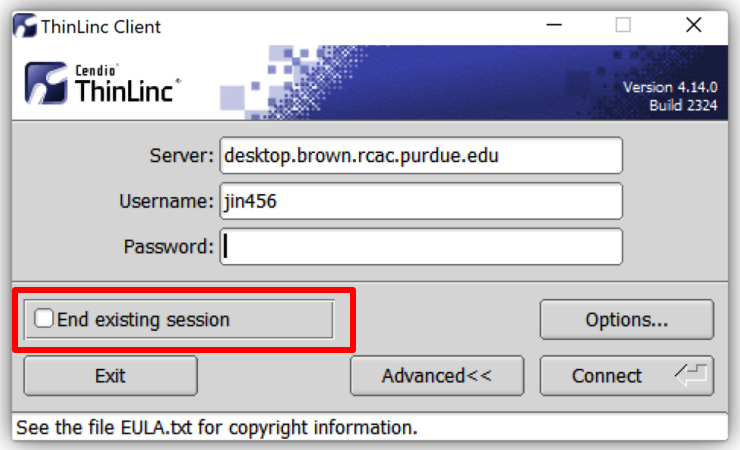
Mac and Linux

- Terminal or Terminal-like apps (e.g. iTerm)
- **ThinLinc** (Allows persistent connection to a remote graphical desktop session.)

Connect using:

```
ssh myusername@brown.rcac.purdue.edu
```

With ThinLinc:
Server: desktop.brown.rcac.purdue.edu



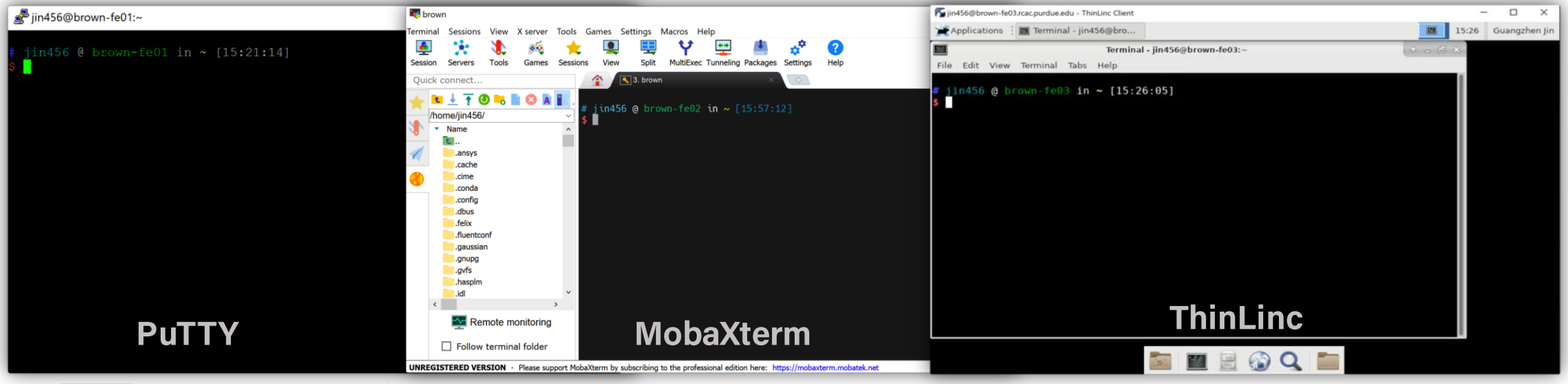
Ref: https://www.rcac.purdue.edu/knowledge/brown/accounts/login?all=true

Connecting - SSH

Windows

Many clients are available for Windows:

- PuTTY
- Windows PowerShell or Windows Subsystem for Linux
- **MobaXterm** (Full-featured SSH client includes X11 support for remote displays, SFTP capabilities, etc.)
- **ThinLinc** (Allows persistent connection to a remote graphical desktop session.)



Connecting - SSH

SSH Keys

- Getting tired of doing 2FA every time? Set up your SSH Key!

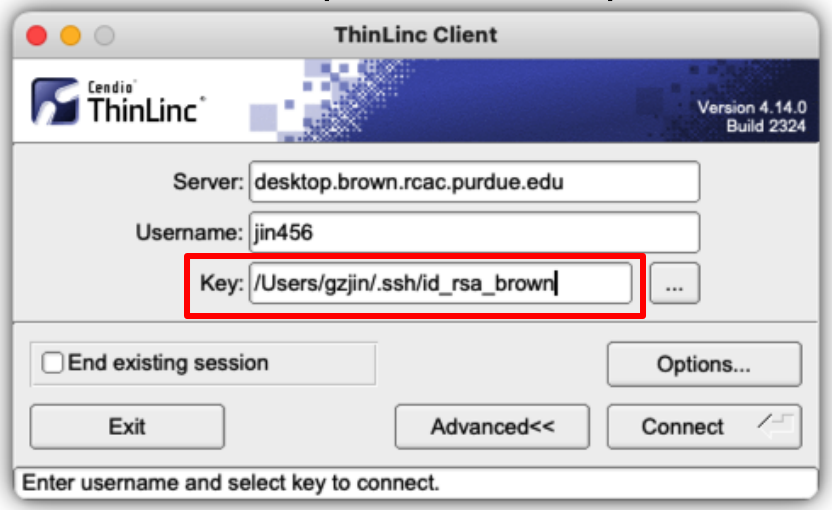
1. Generate a key pair (private and public) on your local machine.
2. Copy the public key to the cluster and append it to `$HOME/.ssh/authorized_keys` file in your account.

Connect without going through 2FA:

```
ssh myusername@brown.rcac.purdue.edu
```

With ThinLinc:

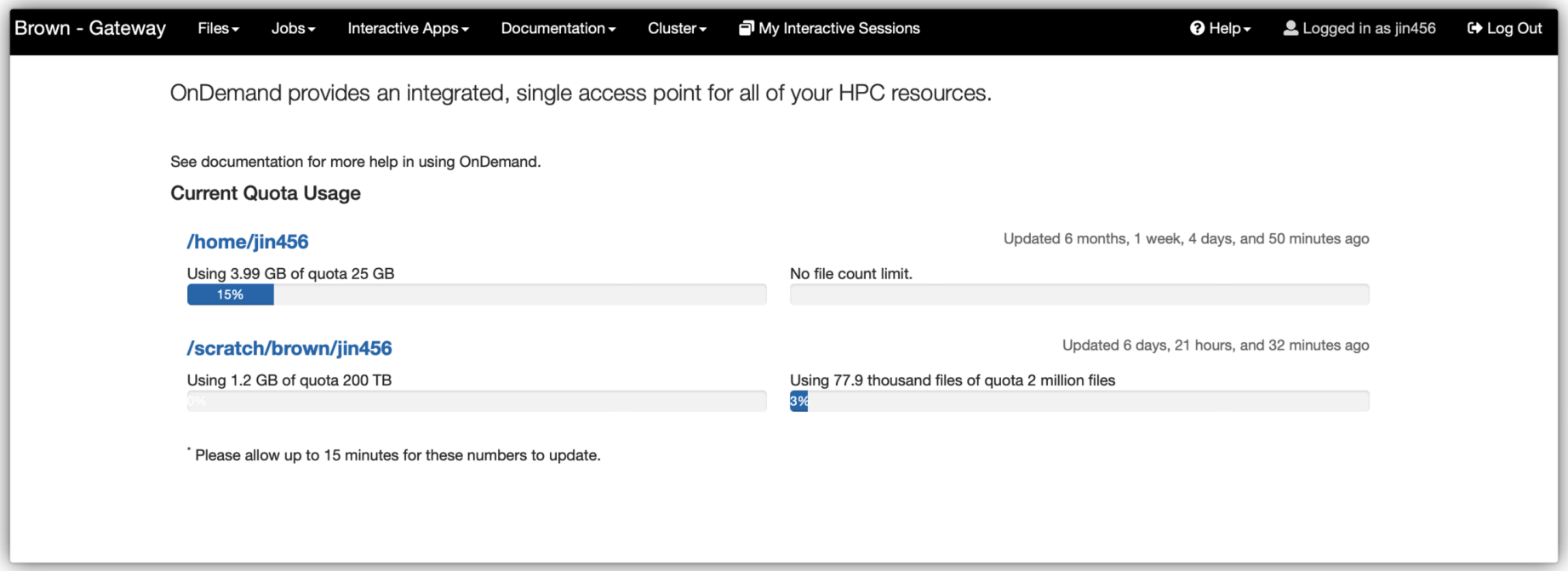
Server: desktop.brown.rcac.purdue.edu



Connecting – Open Ondemand

Gateway (Open Ondemand) on Clusters

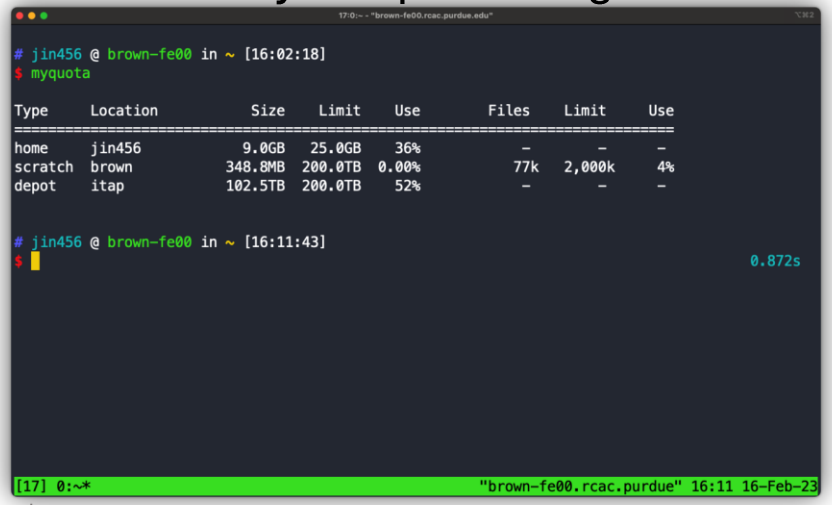
- Navigate to https://gateway.brown.rcac.purdue.edu/



File Systems on Clusters

- Home Directory
  - Personal, long-term file storage. Full backup.
- Scratch Space
  - High-performance, **Short-term** file storage. No backup.
- Depot Space
  - Group-shared, long-term file storage. Full backup.

Use `myquota` command to check your quota usage.



Ref: https://www.rcac.purdue.edu/knowledge/brown/storage/options?all=true

Modules

All RCAC clusters use a module system

- Module system provides for the dynamic modification of a user's environment
- Module commands allow you to add applications and libraries to your environment
- This allows us to simultaneously and safely provide several versions of the same software
- All RCAC clusters have a default programming environment (compiler and MPI library) loaded for you when you log in.

Ref: https://www.rcac.purdue.edu/knowledge/brown/app?all=true

Modules

Useful Commands

- Show all modules currently loaded in my environment:

```
$ module list
Currently Loaded Modules:
  1) intel/17.0.1.132   2) impi/2017.1.132   3) xalt/1.1.2 (S)   4) rcac
```

- Load/unload a module:

```
$ module load matlab
$ module list
Currently Loaded Modules:
  1) intel/17.0.1.132   2) impi/2017.1.132   3) xalt/1.1.2 (S)   4) rcac   5) matlab/R2019a
$ module unload matlab
$ module list
Currently Loaded Modules:
  1) intel/17.0.1.132   2) impi/2017.1.132   3) xalt/1.1.2 (S)   4) rcac
```

- Unload and purge all currently loaded modules:

```
$ module purge
```

Modules

Useful Commands

- Show all available software modules on cluster:

```
$ module avail
```

- Filter available software (e.g. matlab)

```
$ module avail matlab
---------------------------------- Core Applications ----------------------------------
   matlab/R2017a    matlab/R2019a (D)    matlab/R2020b    matlab/R2022a
   matlab/R2018a    matlab/R2020a        matlab/R2021b
```

Note: (D) = default. This software version will be loaded if you don't specify a version.

Useful Commands

- Find software not immediately available
  - Some apps are built with dependencies. When dependencies are not loaded, the app can not be seen.

```
$ module avail openmpi
Lmod has detected the following error:  These module(s) exist but cannot be loaded
as requested: "openmpi"
   Try: "module spider openmpi" to see how to load the module(s).
$ module spider openmpi
-------------------------------------------------------------------------------------
  openmpi:
-------------------------------------------------------------------------------------
     Versions:
        openmpi/1.10.7
        openmpi/2.1.6
        openmpi/3.1.4
-------------------------------------------------------------------------------------
  For detailed information about a specific "openmpi" module (including how to load the modules) use the module's full
name.
  For example:
     $ module spider openmpi/3.1.4
```

Useful Commands

- Find software not immediately available

```
$ module spider openmpi/3.1.4
----------------------------------------------------------------------------------------
  openmpi: openmpi/3.1.4
----------------------------------------------------------------------------------------
    You will need to load all module(s) on any one of the lines below before the "openmpi/3.1.4" module is available to
load.
      gcc/4.8.5
      gcc/5.2.0
      gcc/6.3.0
      gcc/7.3.0
      gcc/8.3.0
      intel/16.0.1.150
      intel/17.0.1.132
      intel/18.0.1.163

$ module load gcc/8.3.0 openmpi/3.1.4
```

Modules

What does module load do under the hood?

- It will change the environmental variables defined for that module. Typically, these are `$PATH` and `$LD_LIBRARY_PATH`. Also a few extra module-specific environmental variables could be set.

```
$ module show matlab
-------------------------------------------------------------------------------
   /opt/spack/modulefiles/Core/matlab/R2019a.lua:
-------------------------------------------------------------------------------
whatis("Name : matlab")
whatis("Version : R2019a")
… …
setenv("MATLAB_HOME","/apps/spack/brown/apps/matlab/R2019a-gcc-4.8.5-jg35hvf")
setenv("RCAC_MATLAB_ROOT","/apps/spack/brown/apps/matlab/R2019a-gcc-4.8.5-jg35hvf")
… …
append_path("PATH","/apps/spack/brown/apps/matlab/R2019a-gcc-4.8.5-
jg35hvf/bin/glnxa64:/apps/spack/brown/apps/matlab/R2019a-gcc-4.8.5-jg35hvf/bin")
append_path("LD_LIBRARY_PATH","/apps/spack/brown/apps/matlab/R2019a-gcc-4.8.5-jg35hvf/runtime/g
lnxa64:/apps/spack/brown/apps/matlab/R2019a-gcc-4.8.5-jg35hvf/bin/glnxa64")
```

Modules

A few tips from us:

- Don't load modules in .bashrc/.profile/.login/.cshrc.
- Don't load more than needed for current task.
- Do `module purge` when in doubt.
- Do check with `module show` what a given module does.
- Do check with `module list` what's actually loaded.
- Do script and automate your builds - even if it's just a 3-line snippet! Reproducibility is good!

Jobs, Queues and SLURM

What is a job?

- A job is simply a set of tasks to be performed by a cluster.
- A script to instruct the cluster precisely what to do to complete your work.
- Self contained to be executed without any interaction.
- Submit and forget!

Where should a job go?

- Always remember: cluster front-ends are shared resources for all login users. Do not run heavy jobs on front-ends.
- Instead, we want jobs carefully arranged on compute nodes.

What happens after a job is submitted?

- Cluster executes jobs on back-end compute nodes.
- Jobs are carefully scheduled and arranged on the compute nodes.

Ref: https://www.rcac.purdue.edu/knowledge/brown/run?all=true

Jobs, Queues and SLURM

Jobs need to specify the resources they require:

- Three basic units:
  - ❑ Number of nodes
  - ❑ Number of cores
  - ❑ Wall Time
- Memory
- Other resources (e.g. GPU)

```
#!/bin/bash
#SBATCH -N 1
#SBATCH -n 2
#SBATCH -A standby
#SBATCH -t 5:00:00
#SBATCH --mem=128G
#SBATCH -o test.out
#SBATCH -e test.err

# Environment setup
… …
# Module load
… …
# Run your job
… …
```

A sample of job submission file

What will the cluster do to you jobs?

- Cluster will allocate requested resources once they are available. Job starts once resources are allocated.
- Cluster allows your job to use only the resources you request.
- Cluster will make sure your job does not run over wall time.

Jobs, Queues and SLURM

Why there are queues on a cluster?

- Cluster can't run all jobs at once so sometimes you must wait
  - Jobs are submitted into queues
  - Jobs wait until cluster and queue has free resources
  - Queues set constraints on jobs that can be submitted
    - ❑Max nodes
    - ❑Max walltime
  - Sets initial priorities
  - Clusters have several queue types
- Check you available queues with `slist`

```
$ slist
                    Current Number of Cores                          Node
Account           Total    Queue     Run    Free    Max Walltime    Type
==============  ================================  ==============  ======
debug                48      401      24      24        00:30:00       A
standby           12456   229766    6104     276        04:00:00       A
```

Jobs, Queues and SLURM

Some facts about cluster queues

- There are generally 3 kinds of queues on a cluster
  - ❑ **Owner queues** (Max walltime=14 days)
  - ❑ **Standby queues** (Max walltime=4 hours)
  - ❑ **Debug queues** (Max walltime=30 mins)
- **Owner queues** are **not** tied to specific nodes in the cluster. They allow you to use the number of nodes somewhere in the cluster.
- Jobs on **Owner queues** can *usually* start within 4 hour.
- **Standby queues** use idle nodes from Owner queues.
- Everyone on cluster gets access to **Standby queues**.
- Lowest priority jobs on **Standby queues**, no promises on turnaround time (can be minutes or days).
- **Debug queues** allow for running small jobs for testing and debugging.
- Jobs on **Debug queues** have highest priority, so can usually start within minutes.

Ref: https://www.rcac.purdue.edu/knowledge/brown/run?all=true

Jobs, Queues and SLURM

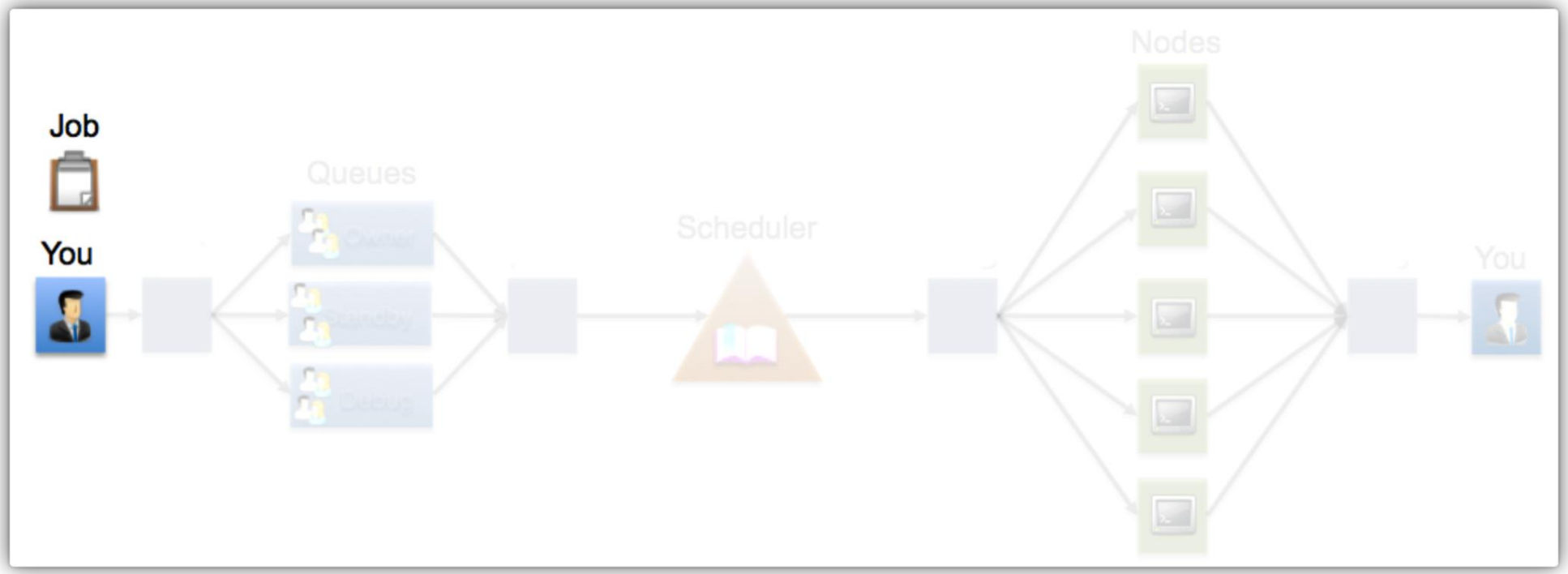
What is SLURM?

- SLURM (Simple Linux Utility for Resource Management) is a system providing job scheduling and job management on compute clusters.
- With SLURM, a user requests resources and submits a job to a queue. The system will then take jobs from queues, allocate the necessary nodes, and execute them.
- SLURM will manage cluster resources for:
  - ❑ Job submission
  - ❑ Job status
  - ❑ Schedules jobs
  - ❑ Executes jobs
  - ❑ Manages queues
  - ❑ Manages nodes

Ref: https://www.rcac.purdue.edu/knowledge/brown/run?all=true

Life of a job

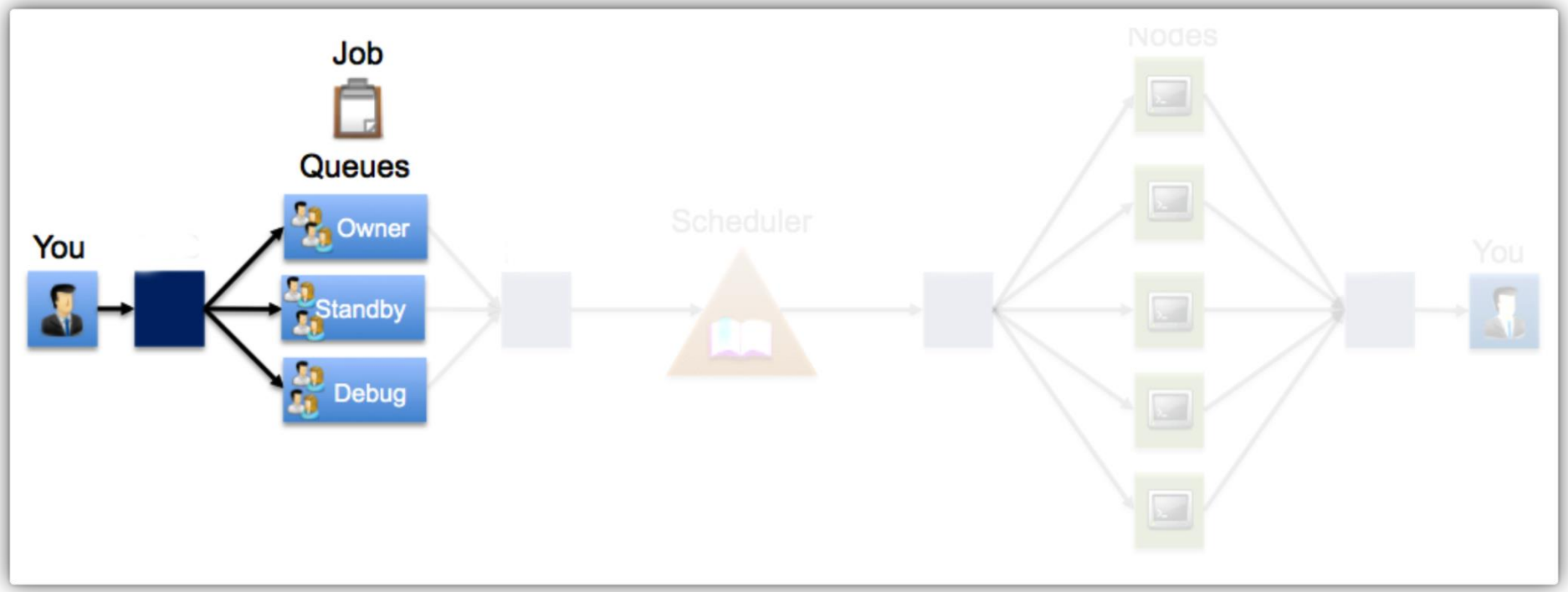
You prepare a job submission script



Ref: https://www.rcac.purdue.edu/knowledge/brown/run?all=true

Life of a job

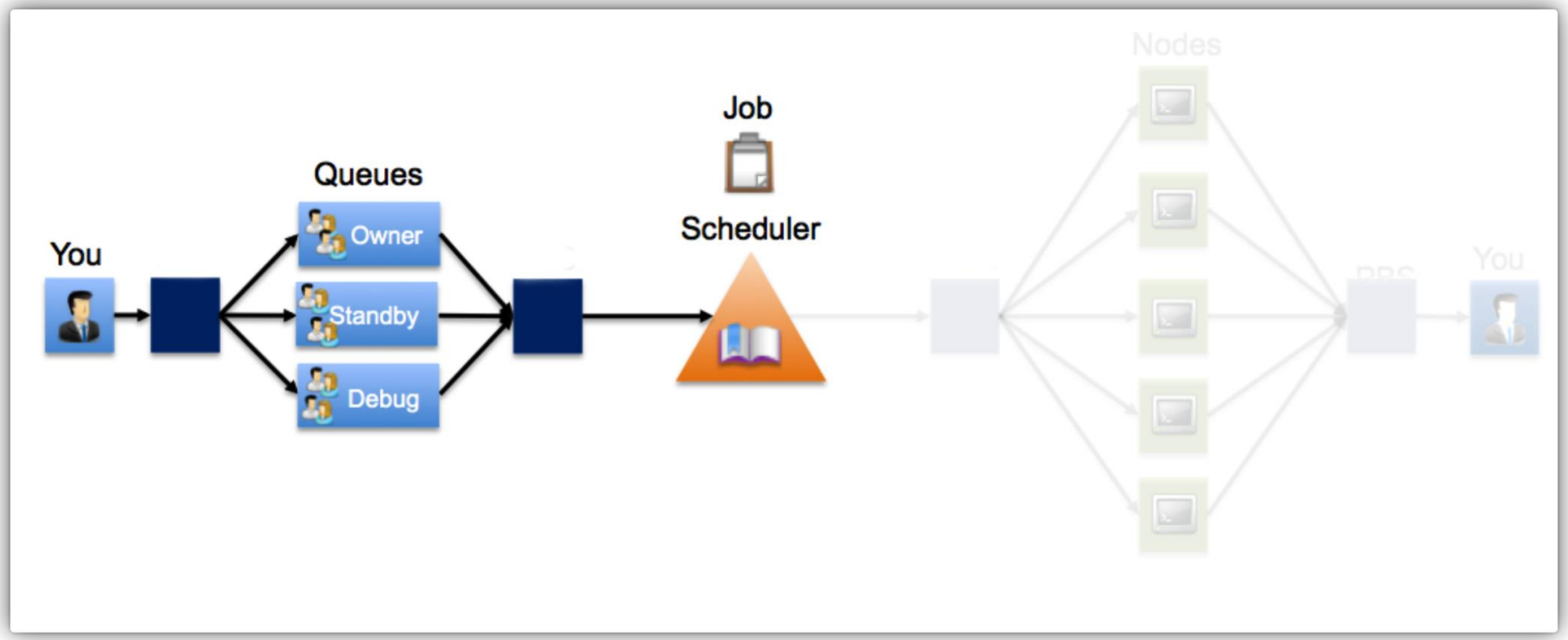
You submit job script into SLURM. The job is placed into a queue.



Ref: https://www.rcac.purdue.edu/knowledge/brown/run?all=true

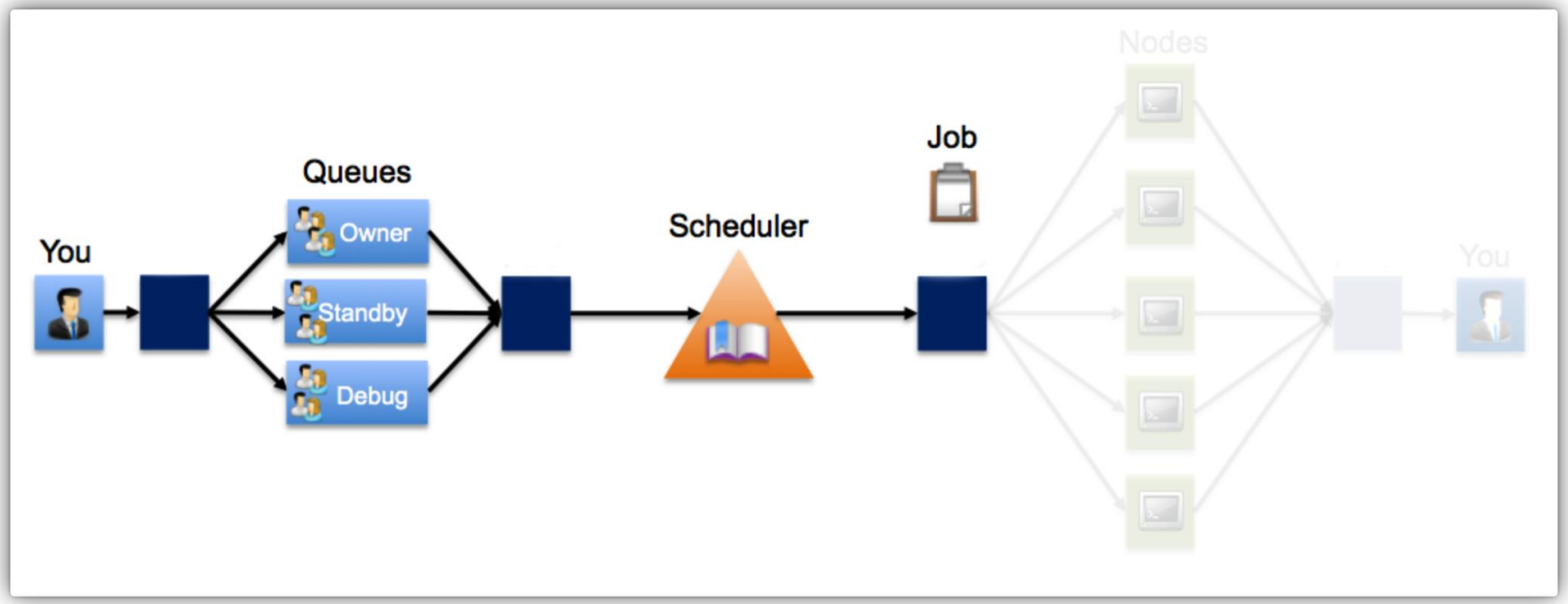
Life of a job

Scheduler iteratively asks for new jobs, status of old jobs, and status of nodes. Each cycle can last several minutes.



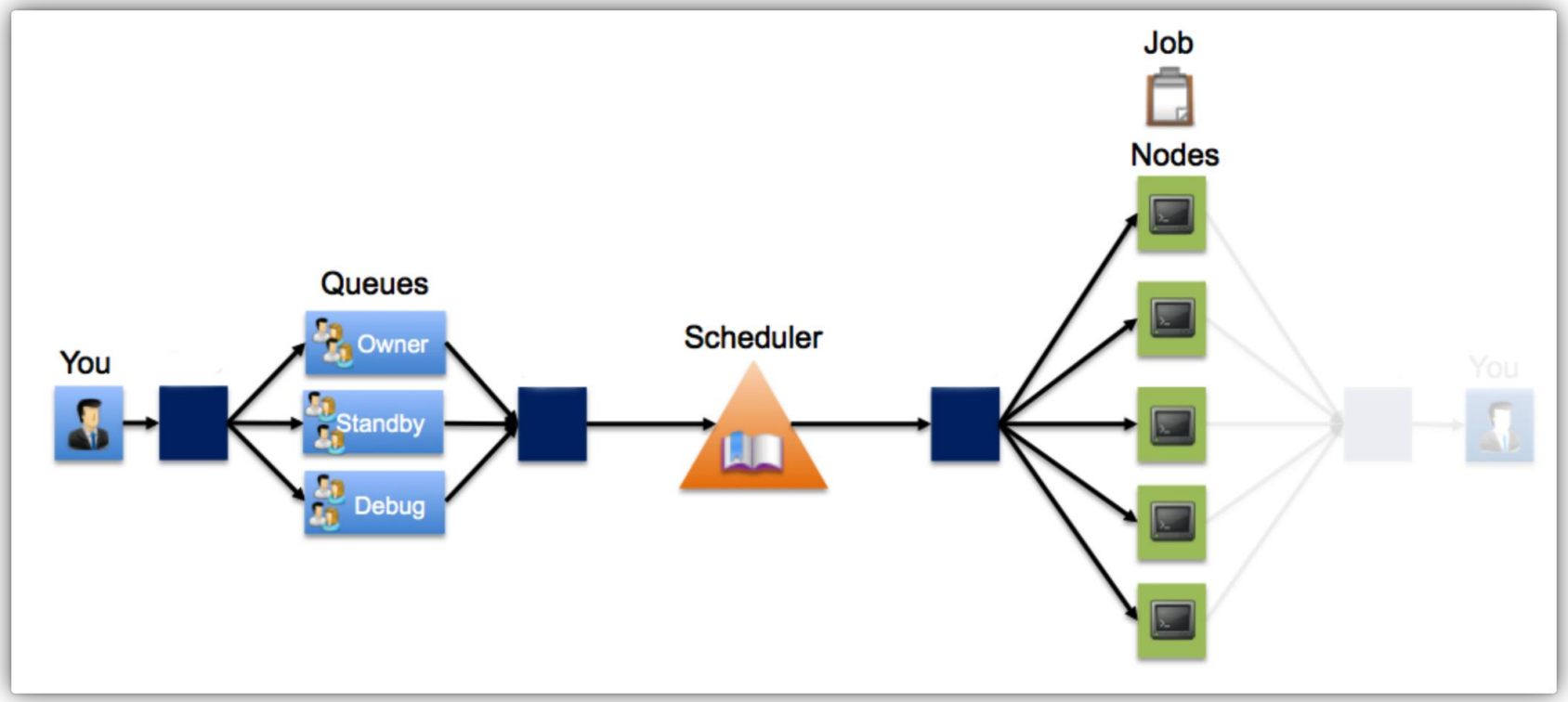
Life of a job

Scheduler considers jobs, acts as a handler. Tells SLURM to start jobs one at a time. Goes back and does this over and over.



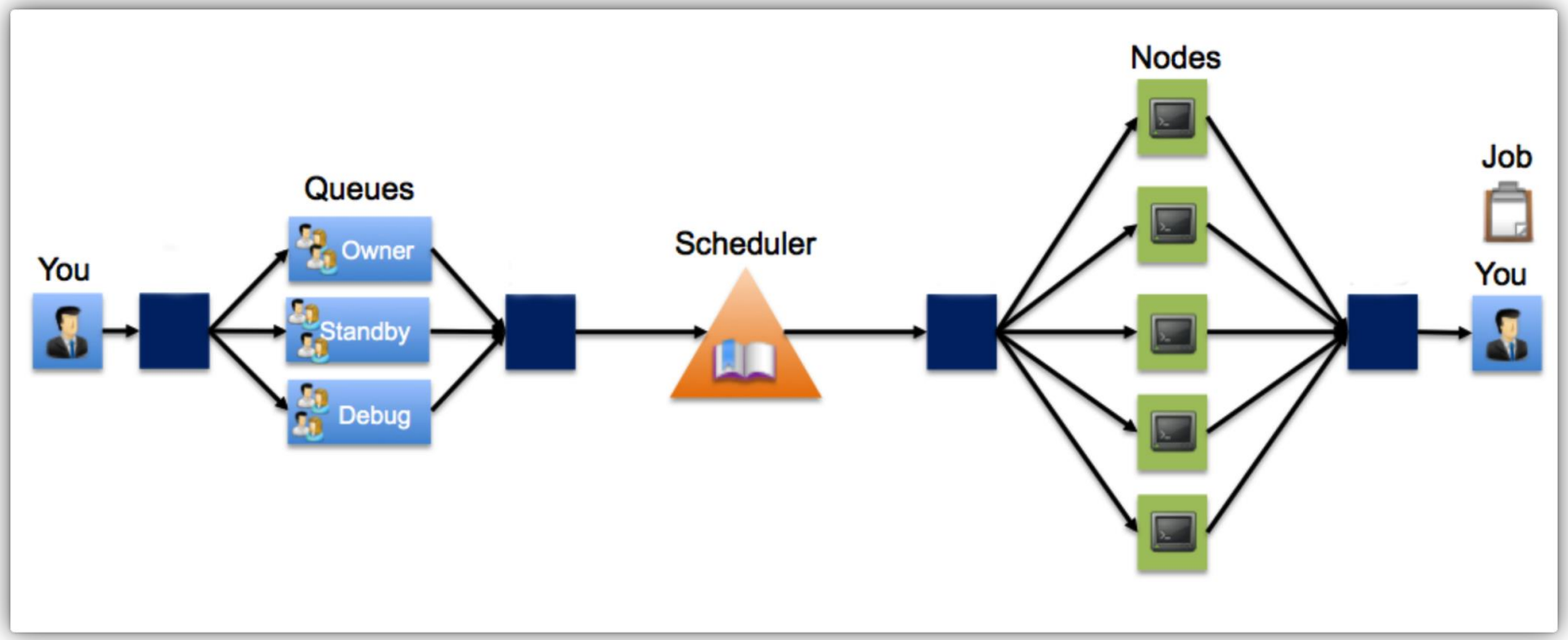
Life of a job

SLURM checks health of the node. Send job script to node(s) and executes it.



Life of a job

SLURM watches for the job script to complete on nodes. Collects job output files from nodes and sends files to you.

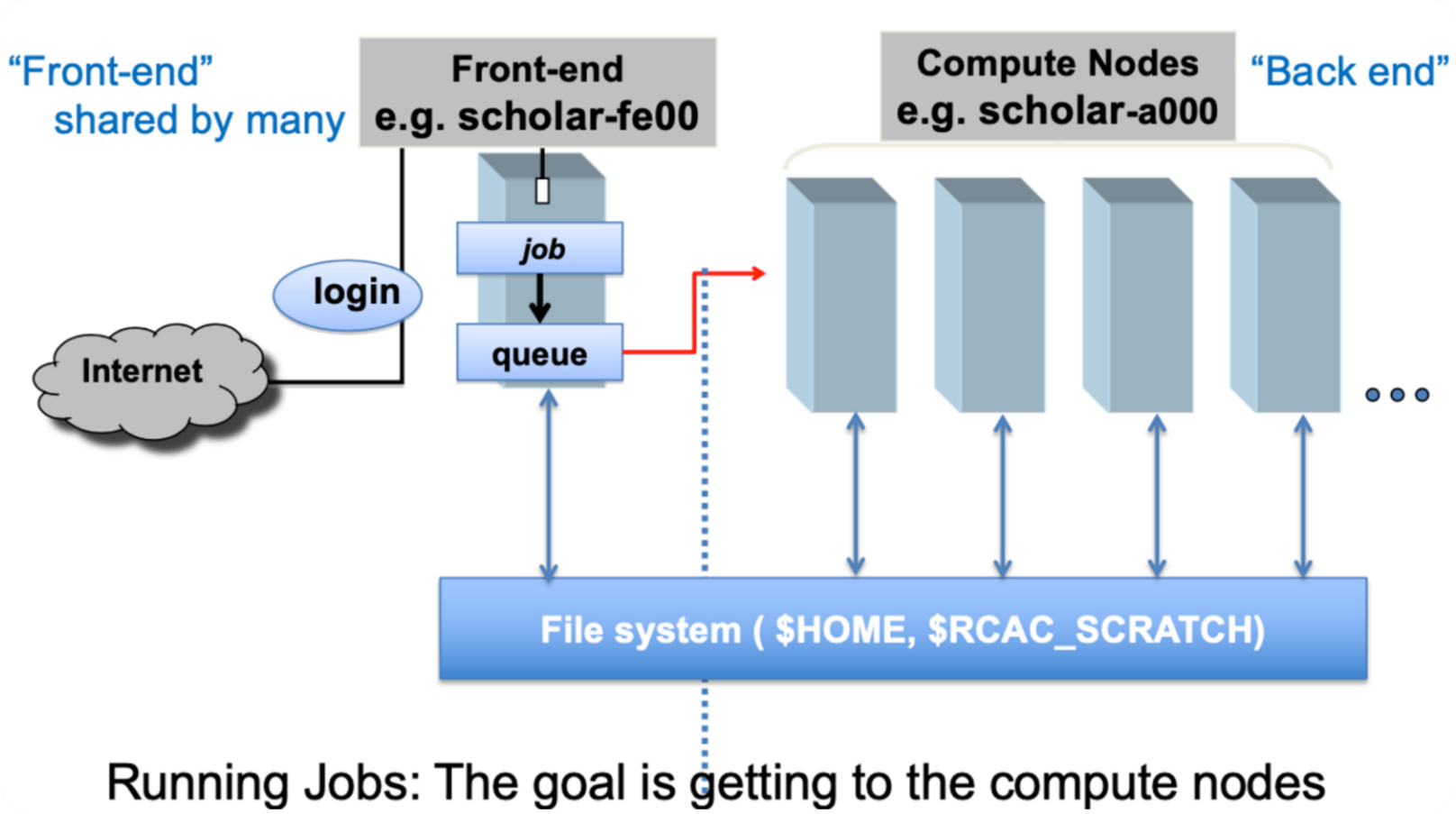


Good Citizenship

Why Good Citizenship is necessary on a community cluster?

- **Resource utilization:** our community clusters are shared resources in terms of processors, memory, storage, and network bandwidth by all users. Good citizenship means using these resources efficiently and avoiding impacting the performance of the entire system.
- **Teamwork:** HPC clusters are often used for collaborative research projects and groups, and good citizenship means working with others in a professional and respectful manner.
- **Security**: Good citizenship means taking responsibility for the security of the system, avoiding activities that could compromise the security of the system or the data stored on it.
- More … …

Front-end vs Compute Node



Good Citizenship on Front-ends

- Front-ends are very limited resources and you'll annoy the system administrators and other users
- Watchdog and we will monitor and quickly terminate (or severely throttle) any inappropriate resource-intensive process
- Do not do major science on the front end. Front-ends can be used for:
  - ❑ Building/compiling software
  - ❑ Managing files: editing, transferring, tar, gzip, hsi
  - ❑ Submitting, monitoring, and managing batch jobs
  - ❑ Launching interactive jobs
  - ❑ Modest post-processing and analysis
- Instead: submit a job to compute nodes.

Good Citizenship on jobs and queues

- Do not request for excessive resources knowingly (e.g. asking for a large memory node when it's not needed)
- Do not abuse file systems (e.g. heavy I/O for /depot space, use /scratch instead)
- Do not submit lots of tiny jobs, use pilot-job pattern with tool.
- Do not submit jobs and camp. (e.g. submitting GPU job from OoD for 24 hours so it's ready for you in the afternoon, and then forget about it)

THANK YOU

Let us know if you have further questions by:

1) Sending us tickets via rcac-help@purdue.edu;
2) Joining online or in-person Coffee Hours
   via https://www.rcac.purdue.edu/coffee

Q&A

CLUSTERS 101

For more advanced cluster usage like job submission and management, welcome to our Clusters 201 next Friday!