CLUSTERS 201

Guangzhen Jin

Senior Computational Scientist

PURDUE UNIVERSITY | Rosen Center for Advanced Computing

Outlines

- Introductions of clusters
- Front-ends vs Compute nodes
- Interactive jobs vs Batch job
- Submit/Monitor/View jobs
- Fortress - HSI & HTAR

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.

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:

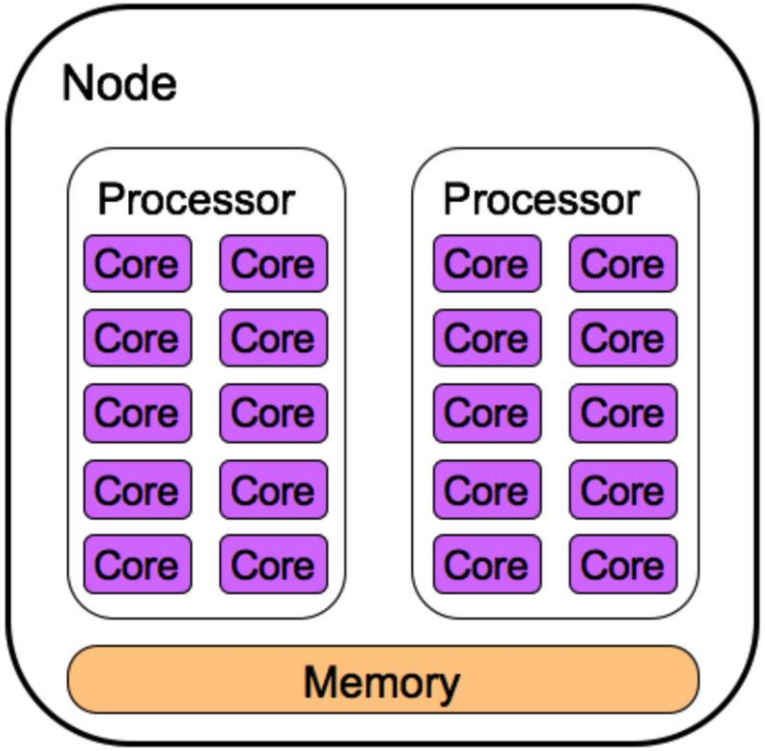
- 2 physical processors, 10 cores ea.

Queuing system sees:

- 20 logical processors

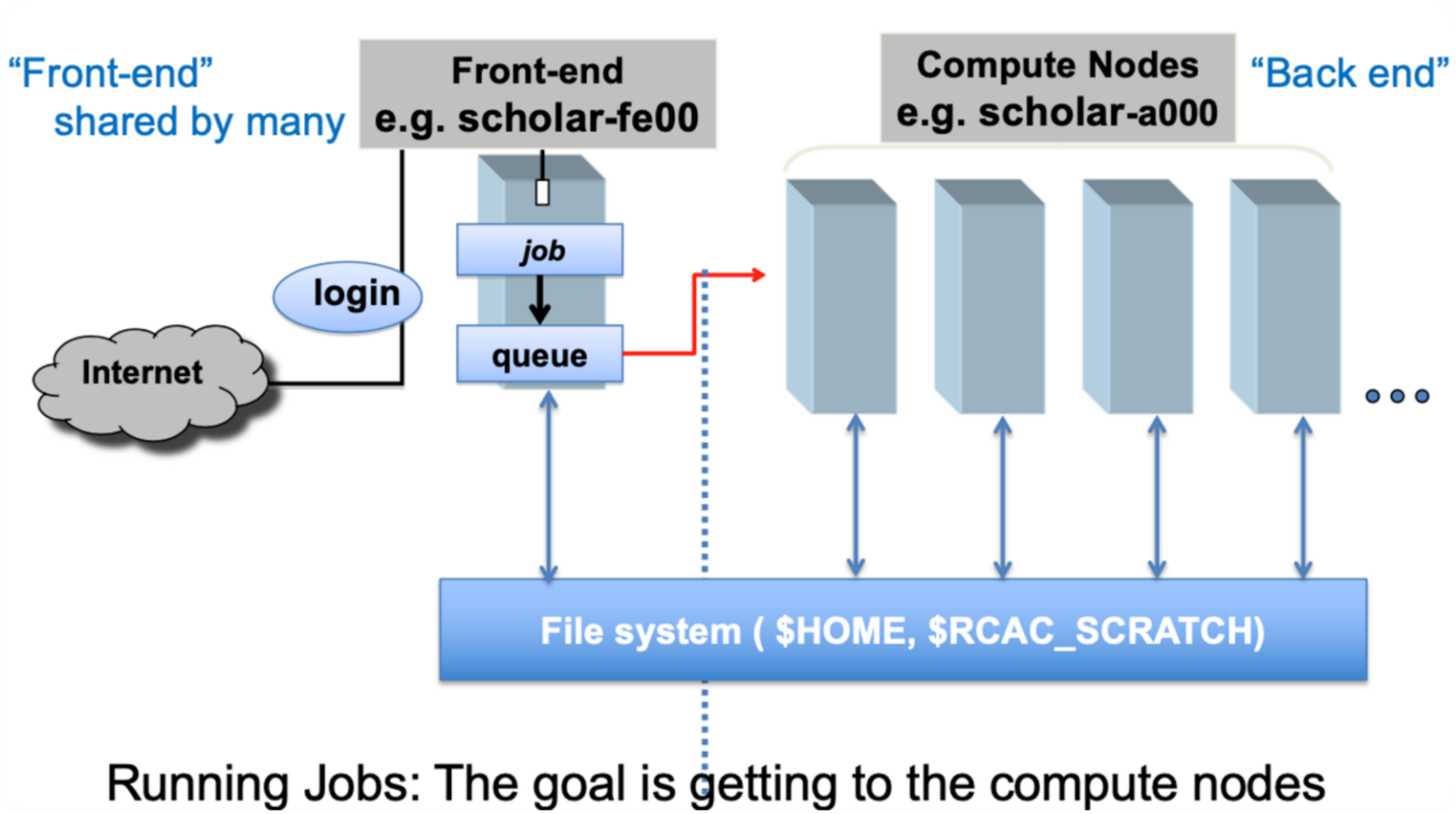
From now on, we will be mostly concerned with cores (logical processors), not physical chips

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



Front-ends vs Compute nodes

Front-end vs Compute Node



Front-ends vs Compute nodes

Font-end: Where NOT to run a job

- Remember, cluster front-end nodes are shared resources for
  - Creating, submitting, and monitoring jobs
  - File transfers
  - Preparing inputs
  - Editing and compiling code
  - Small-scale testing

- **Do not do science on the front end!**

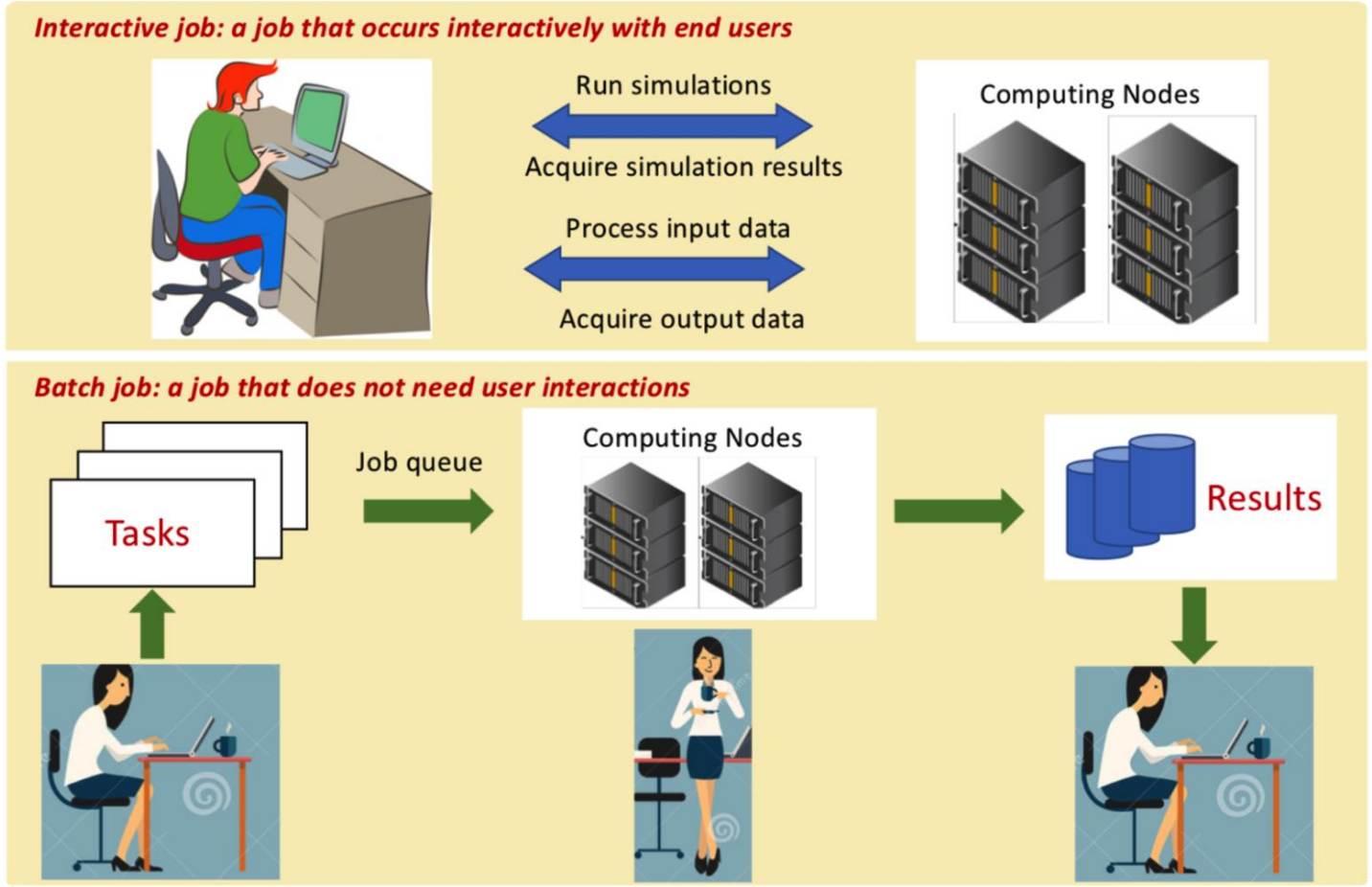
Front-ends vs Compute nodes

Compute Node: Where to run a job

- Instead: grab a compute node
- Cluster executes jobs on back-end compute nodes
- Jobs are carefully scheduled and arranged on the compute nodes
- Interactive vs batch job

Interactive jobs vs Batch jobs

Interactive Job vs Batch Job



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)

Job submission and management

Before job submissions:

Basic tools to check your account/job status

- Check cluster features: `sfeatures`
- Check queue availability: `slist username`
- Check current jobs: `squeue -u username`

  `squeue -A accountname`
- Check previous jobs: `sacct -u username -S start_date -E end_date`

Job submission and management

Why an interactive job?

- Dedicated compute node (vs a shared frontend)
- Test code without impacting others
- Quicker develop / test / debug cycle
- Run GUI apps as a job
  - Matlab
  - Fluent
  - Windows VM

Job submission and management

How to run an interactive job?

- Remote Desktop (ThinLinc Web/Client)
  - Application Menu
  - `sinteractive`
- Gateway (Open OnDemand)
- Local terminal (with X11 forwarding if GUI needed)

Job submission and management

Submit an interactive job

```
 # Commands to submit an interactive job (Example)
$ sinteractive -A queue_name -N 1 -n 20 -t 01:00:00
salloc: Granted job allocation 19050398
salloc: Waiting for resource configuration
salloc: Nodes xxx are ready for job
```

```
# Commands to submit an interactive job (Example on Gilbreth)
$ sinteractive -A queue_name -N 1 -n 1 --gres=gpu:1 -t 01:00:00
```

Job submission and management

Submit a batch job

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=10
#SBATCH --account=queue_name
#SBATCH --job-name=mpi_job
#SBATCH --time=01:00:00
#SBATCH --output %x-%u-%j.out

# Module load and environment setup
module purge
module load rcac
module list

# Your jobs goes here
mpirun -np $SLURM_NTASKS ./mpi_hello
```

```
# Submit the job
$ sbatch batch_test.sub
```

Job submission and management

Inside of job submission file

- **SLURM directives**
  - Specify resources needed such as number of nodes, cores
- **Modules and environments**
  - Set up paths, libraries
- **SLURM environment variables**
  - Set by SLURM, can be used in your submission script
- **Customized commands**
  - Your job to run

Job submission and management

Submit a batch job

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=10
#SBATCH --account=queue_name
#SBATCH --job-name=mpi_job
#SBATCH --time=01:00:00
#SBATCH --output %x-%u-%j.out
# Module load and environment setup
module purge
module load rcac
module list
# Your jobs goes here
mpirun -np $SLURM_NTASKS ./mpi_hello
```

SLURM directives

Module & Environment

Commands

Job submission and management

Basic tools to check job status and manage a job

- Check job info: `jobinfo jobid` or `scontrol show job jobid`
- Cancel a job: `scancel jobid`
- Hold a job: `scontrol hold jobid`
- Release a job: `scontrol release jobid`

Monitor resources with `monitor`

- Load the module (`module load utilities monitor`) and use inside of your batch job

Ref: https://www.rcac.purdue.edu/knowledge/brown/run/slurm

Fortress - HSI & HTAR

Fortress tape archive

- Tape library with a robotic arm and a disk cache in front (i.e. fast uptake, slow egress)
- Redundant hardware, never purged, protected by multiple physical copies on separate tapes.
- Accessible from all RCAC clusters as well as on- and off-campus with additional tools
- Huge (25 PB), free and practically unlimited
- Everyone with an RCAC account gets personal Fortress space.
  Additionally, labs with Depot also get lab Fortress space
  Personal and group spaces
  `/home/myusername` and `/group/mylab`

Fortress - HSI & HTAR

Fortress access by `hsi` and `htar`

- `hsi` and `htar` are special command-line utilities for HPSS tape archive
  - Installed on all clusters, available for download for other Linux computers
- `hsi` is remote shell-like interface to Fortress
  - navigate, list, manipulate files, transfer things in or out
  - `ls`, `cd`, `mkdir`, `cp`, `mv`, `put`, `get` – great for interactive work (and can also be batched)

Ref: https://www.rcac.purdue.edu/knowledge/fortress/storage/transfer/hsi

Fortress - HSI & HTAR

Fortress access by hsi and htar

- `htar` closely mimics regular `tar`
  - create, list, extract tarballs stored on Fortress (also makes a matching index file for faster searches)

```
htar -cPvf /path/on/fortress/archive.tar datadir(s)  # store
htar -xvf /path/on/fortress/archive.tar [file(s)]    # extract
htar -tvf /path/on/fortress/archive.tar              # list
```

- `htar` has an individual file size limit of 64GB > `htar_large`
  - e.g. you can use `htar` for a terabyte-size tarball as long as no individual file is larger than 64GB

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 201

Visit our website for more upcoming trainings:
https://www.rcac.purdue.edu/news/events