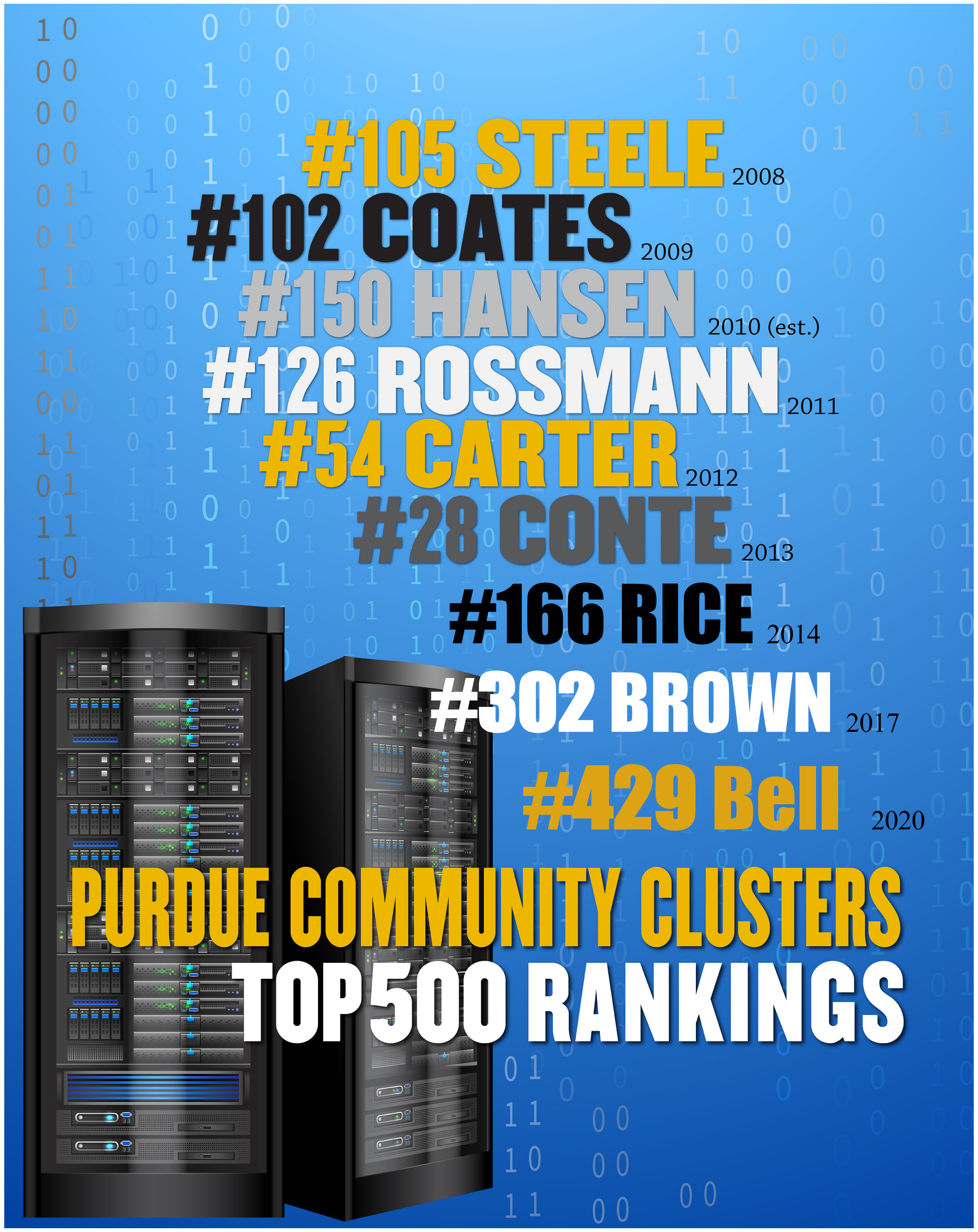
Cluster 201

Xiao Zhu

Research Computing
Purdue

Examples and lab: /depot/itap/training/cluster201

PURDUE'S OWN CLUSTERS



PURDUE'S OWN CLUSTERS

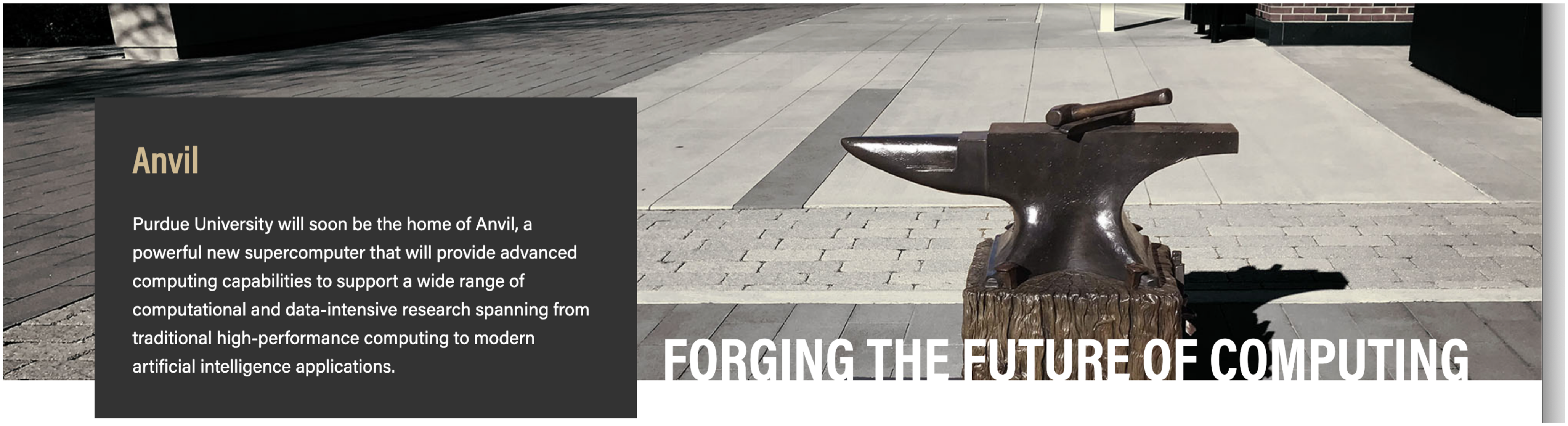
Community clusters:

HPC (Halstead, Brown, Bell): 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 P100 and V100 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.

PURDUE'S NEXT CLUSTER - Anvil



Anvil

Purdue University will soon be the home of Anvil, a powerful new supercomputer that will provide advanced computing capabilities to support a wide range of computational and data-intensive research spanning from traditional high-performance computing to modern artificial intelligence applications.

- 10 million dollar award from National Science Foundation
- enabling important discoveries across many different areas of science and engineering
- serve as an experiential learning laboratory for students to gain real-world experience using computing for their science

PURDUE'S NEXT CLUSTER - Anvil

Anvil Webpage
https://rcac.purdue.edu/anvil

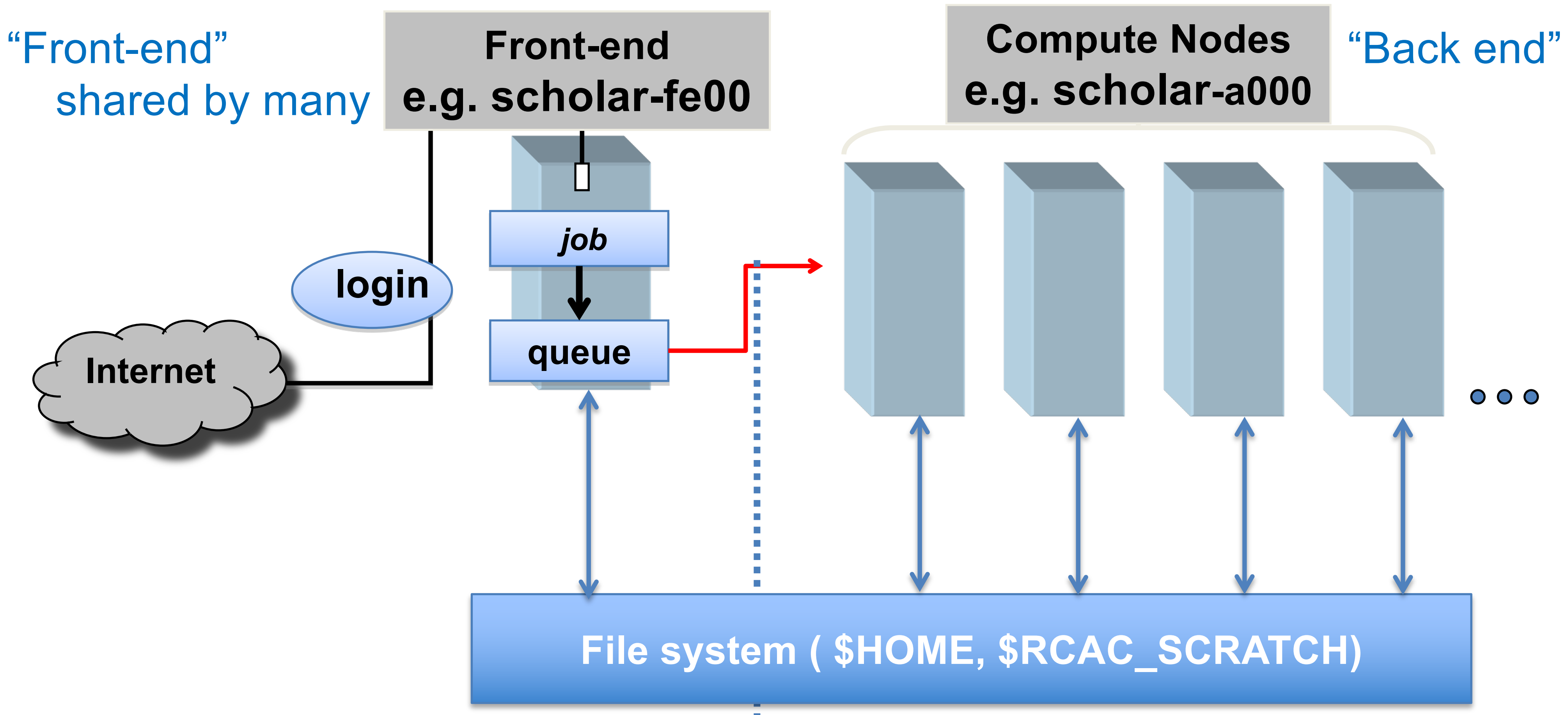
EUP application
https://rcac.purdue.edu/anvil/earlyuser

XSEDE allocation request
https://portal.xsede.org/submit-request

Send questions to:
anvil@purdue.edu

Apr 30th	EUP applications close
Jun 15th – Jul 15th	Anvil available for XSEDE allocation
Aug 01st – Aug 31st	Early User Program
Oct 1st	Anvil enters production

FRONT-END VS COMPUTE NODE



Running Jobs: The goal is getting to the compute nodes

FRONT-ENDS

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
- May be used by 50+ people simultaneously
- **Do not do science on the front end!**
  - Either it's simply not possible
  - Or you'll annoy the system administrators and other users

https://en.wikipedia.org/wiki/Rapid_transit#/media/File:Moscow_MetroCrowded_(pixinn.net).jpg

COMPUTE NODES

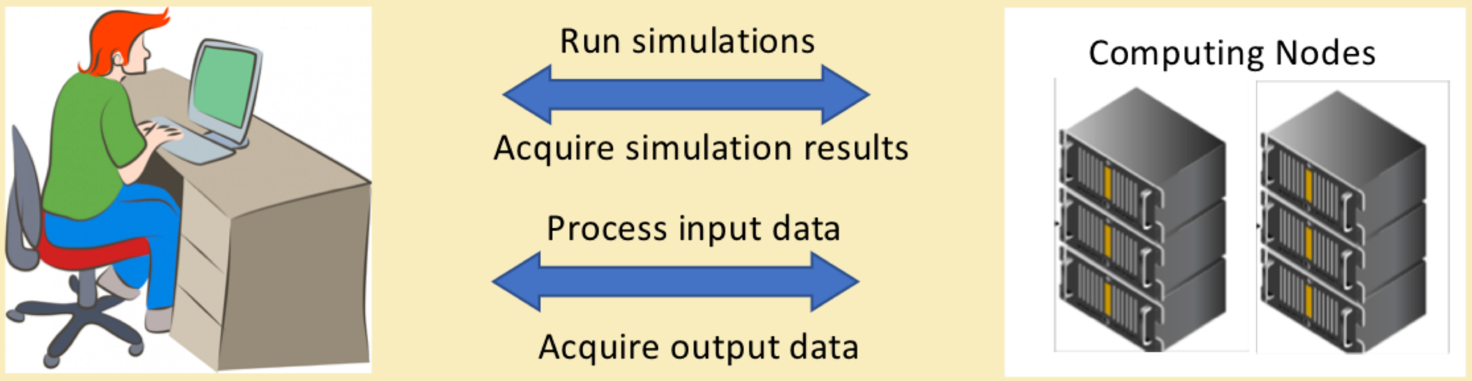
WHERE TO RUN A JOB

- Instead: grab a compute node. We’ll get to that.
- Cluster executes jobs on back-end compute nodes
- Jobs are carefully scheduled and arranged on the compute nodes
- Interactive vs batch job

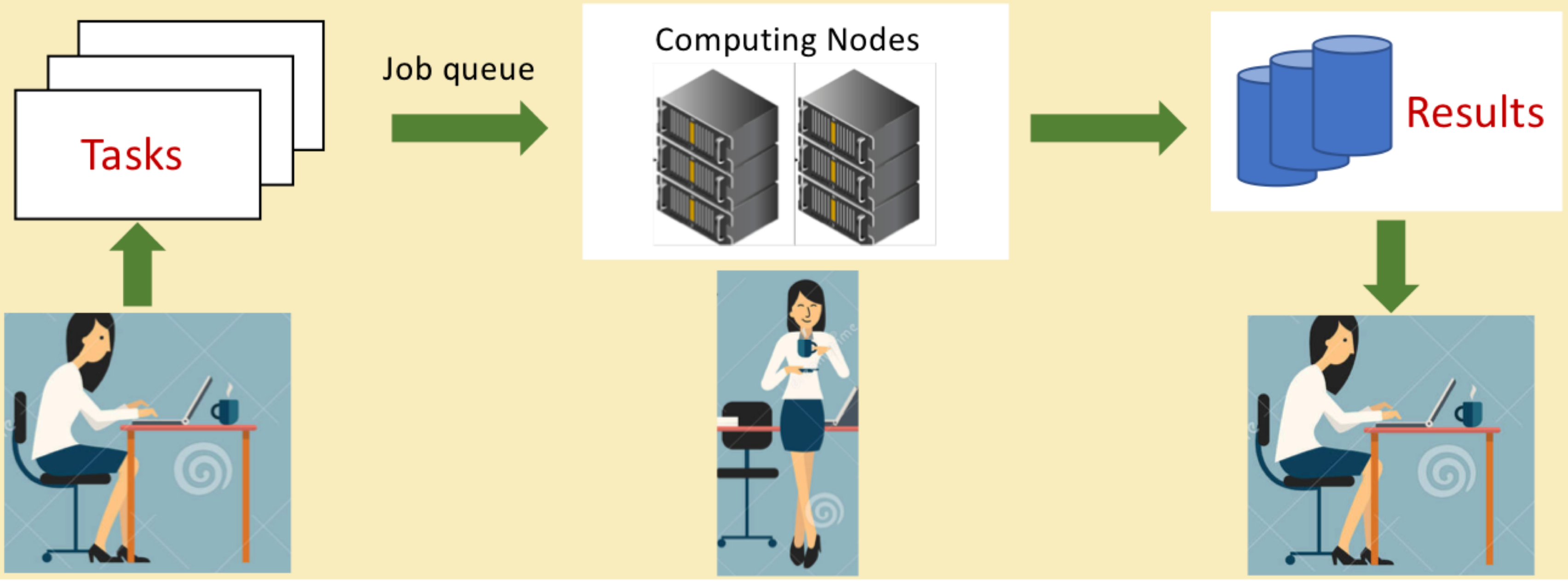
https://en.wikipedia.org/wiki/SNCF_TGV_Duplex#/media/File:TGV_Dupex_First_Class.jpg

INTERACTIVE JOBS

Interactive job: a job that occurs interactively with end users



Batch job: a job that does not need user interactions



JOB SUBMISSION SCRIPT

- Commands that instructs cluster precisely what to do to complete your work
- Self contained to be executed without any interaction
- Jobs need to specify the resources they require
  - Three basic units:
    - Number of nodes
    - Number of cores
    - Time
  - Memory
  - Other resources
- Cluster will allocate requested resources once they are available
- Job starts once resources are allocated

EXAMPLE JOB SUBMISSION SCRIPT

subjob.sub

```
#!/bin/sh -l
#SBATCH -J mpi_job
#SBATCH -A scholar
#SBATCH -N 1
#SBATCH -n 20
#SBATCH -t 03:00:00

module purge
module load rcac
module list

date +"%d %B %Y %H:%M:%S"

# An MPI job
mpirun -np 20./a.out

date +"%d %B %Y %H:%M:%S"
```

COMPONENTS OF A SUBMISSION SCRIPT

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

SLURM DIRECTIVE

- A way to set SLURM job attributes
- Appear at the top of your submission file
- Common SLURM job attributes include:

SLURM directives	Description
`#SBATCH -N 1`	Number of nodes required (--nodes=1)
`#SBATCH -n 20`	Number of tasks required (--ntasks=20)
`#SBATCH -A scholar`	The destination queue of the job
`#SBATCH -J test`	The name of the job
`#SBATCH -t 00:10:00`	The estimated maximum walltime of the job, the job will be killed beyond this walltime
`#SBATCH --mem=60G`	Amount of memory per node requested
`#SBATCH -G 2`	Number of GPUs requested (--gpus=2)

EXAMPLE JOB SUBMISSION SCRIPT

subjob.sub

```
#!/bin/sh -l
#SBATCH -J mpi_job
#SBATCH -A scholar
#SBATCH -N 2
#SBATCH -n 48
#SBATCH -t 03:00:00
#SBATCH -o %x-%u-%j.out

module purge
module load rcac
module list

cd $SLURM_SUBMIT_DIR
pwd

mpirun -np $SLURM_NTASKS a.out
```

SLURM ENVIRONMENT VARIABLES

$SLURM_JOBID: unique SLURM job id

$SLURM_JOB_NAME: job name supplied by the user

$SLURM_SUBMIT_DIR: Absolute path of the current working directory when you submitted this job

$SLURM_NTASKS: number of execution slots (cores) for the job

$SLURM_NTASKS_PER_NODE: number of execution slots per node for the job

$SLURM_JOB_NUM_NODES: number of execution nodes for the job

$SLURM_JOB_NODELIST: a list of nodes assigned to this job

$SLURM_GPUS: number of GPUs for the job

$SLURM_MEM_PER_NODE: requested memory per node in MB for the job

HOW TO SUBMIT AND MONITOR YOUR JOB

sbatch

Once you have a job submission script, you may submit this script to SLURM using the sbatch command. SLURM will find an available compute node or set of compute nodes and run your job there, or leave your job in a queue until some become available.

```
$ sbatch subjob.sub
Submitted batch job 51338

# here the number indicates the job id that SLURM
# assigns to the job, this number can be used to track
# your job.
```

HOW TO MONITOR THE STATUS OF YOUR JOB

squeue –u myusername

List all current jobs from the user *myusername*, where *myusername* is your Purdue login.

```
 $ squeue -u zhu472


JOBID       USER     ACCOUNT      NAME       NODES    CPUS   TIME_LIMIT ST TIME
51338       zhu472   scholar      testjob        1       4        10:00  R 0:01
```

squeue -A myaccount

squeue -u myusername -O jobid,starttime

HOW TO SUBMIT AND MONITOR YOUR JOB

scancel

Stop and delete the job ID *myjobid*

```
 $ squeue -u zhu472

JOBID     USER     ACCOUNT     NAME      NODES     CPUS    TIME_LIMIT ST TIME
51338     zhu472   scholar     testjob       1        4         10:00  R 0:01

 $ scancel 51338
 $ squeue -u zhu472

JOBID     USER     ACCOUNT     NAME      NODES     CPUS    TIME_LIMIT ST TIME
51338     zhu472   scholar     testjob       1        4         10:00 CG 0:11
```

HOW TO CHECK FOR AVAILABLE RESOURCES

slist

List all queues I can use and their current status and limits.

```
$ slist

                    Current Number of Cores

 Queue              Total  Queue   Run     Free      Max Walltime
 ================  ===================================  ==============
 debug                20     0       0      20          0:30:00
 long                100     0      20      80         72:00:00
 scholar             160     0      20     120          4:00:00
```

HOW TO OBTAIN DETAILED JOB INFORMATION

jobinfo

show you the full history of a job, for both completed jobs and jobs in progress

```
$ jobinfo 2189815
Name                    : job.sub
User                    : zhu472
Account                 : standby
Partition               : bell-standby
Nodes                   : bell-a056
Cores                   : 128
GPUs                    : 0
State                   : COMPLETED
ExitCode                : 0:0
Submit                  : 2021-04-08T10:14:38
Start                   : 2021-04-08T10:15:17
End                     : 2021-04-08T10:15:38
Waited                  : 00:00:39
Reserved walltime       : 04:00:00
Used walltime           : 00:00:21
Used CPU time           : 00:00:02
% User (Computation): 67.73%
% System (I/O)       :  0.00%
Mem reserved            : 1992M/core
Max Mem used            : 3.73M (bell-a056)
Max Disk Write          : 0.00  (bell-a056)
Max Disk Read           : 0.00  (bell-a056)
```

EXERCISE

In the `batchjob` directory:

- Edit the batch job script `job.sub`
    - Use the standby queue
    - Use 2 node
    - Use 40 cores

- Submit the job and check the job status

INTERACTIVE JOBS

WHY RUN INTERACTIVELY ON A COMPUTE NODE?

- 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

INTERACTIVE JOBS

WHY RUN INTERACTIVELY ON A COMPUTE NODE?

- Remote Desktop (ThinLinc)
  - Application Menu
  - sinteractive

- Gateway (Open OnDemand)