



Running Job on RCC systems using the SLURM Scheduler

Raj Shukla and Parmanand Sinha

July 21, 2020

Agenda

- The RCC Midway compute systems
- Using Slurm (Simple Linux Utility for Resource Management) to submit jobs to the RCC Midway systems

=> Goals of this workshop

- Learn the basics of Slurm
- Be comfortable submitting jobs on a Midway HPC systems
- Understand queue priority
- Be comfortable checking if jobs succeeded/failed
- Know how to fix common errors



THE UNIVERSITY OF
CHICAGO

Office of Research and
National Laboratories
Research Computing Center

Understanding the RCC Compute Ecosystem

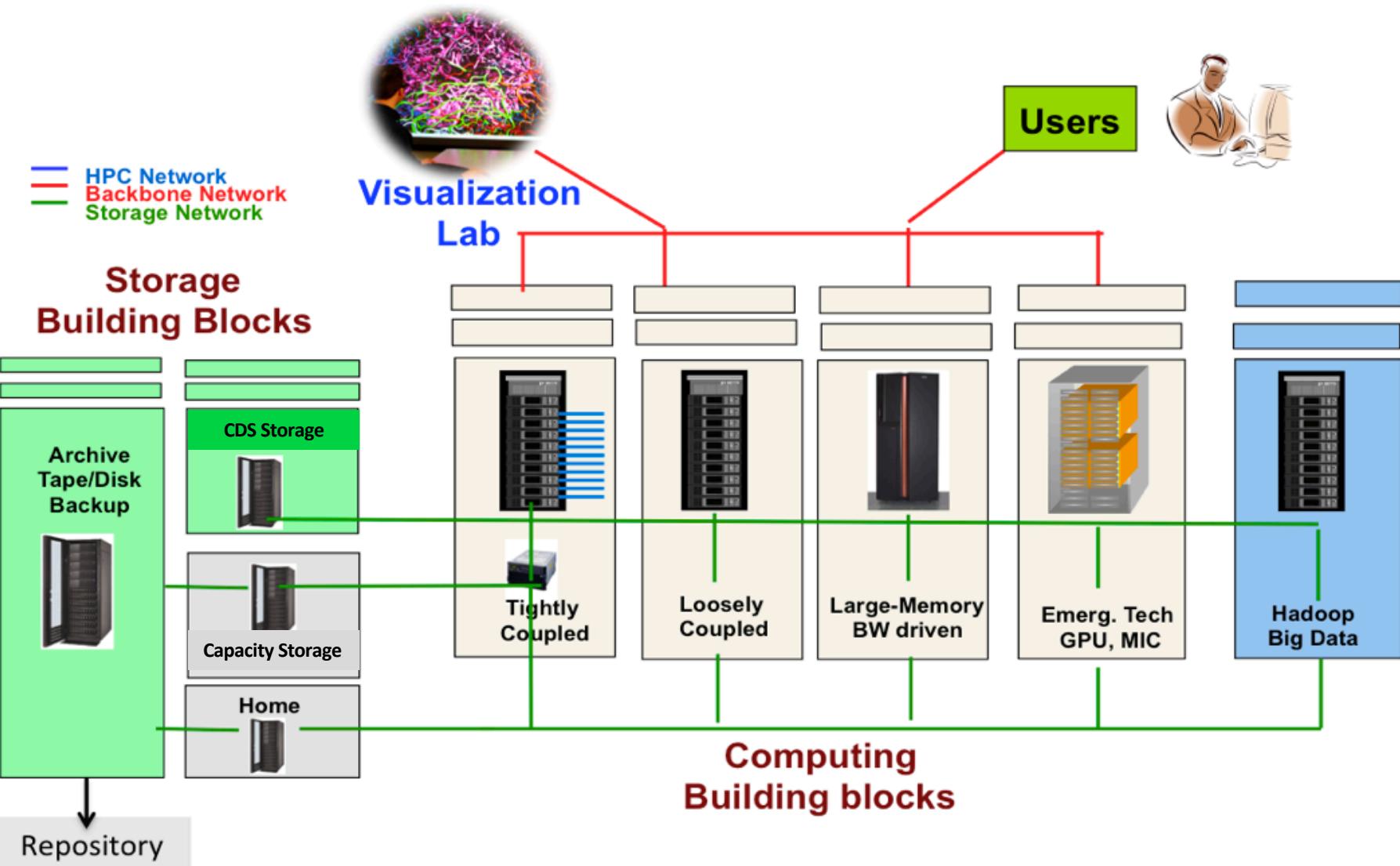
Midway is a constellation of many compute systems and storage with various architectures coupled together in one system.



Slurm is the software used to manage the workload on Midway

Let us begin...

Hardware Infrastructure building blocks



Some definitions

- A **processor** is a small chip that responds to and processes the basic instructions that drive a computer. The term *processor* is used interchangeably with the term **central processing unit (CPU)**
- **Core:** The smallest compute unit that can run a program
- **Socket:** A compute unit, packaged as one and usually made of a single chip often called processor. Modern sockets carry many cores (10, 14, or 20, 24, 28, etc. on most servers)
- **Node:** A stand-alone computer system that contains one or more sockets, memory, storage, etc. connected to other nodes via a fast network interconnect.



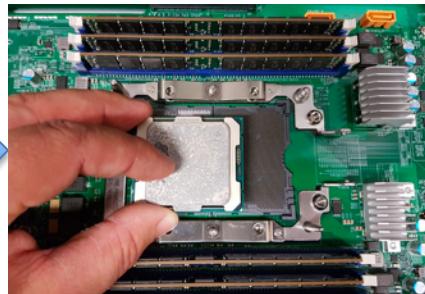
THE UNIVERSITY OF
CHICAGO

Office of Research and
National Laboratories
Research Computing Center

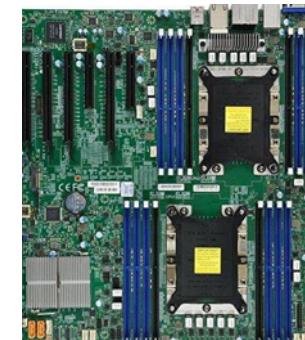
From a CPU to a Cluster



CPU
(Has Multiple cores)



Socket
(Can have Multiple CPUs)



Motherboard



Cluster

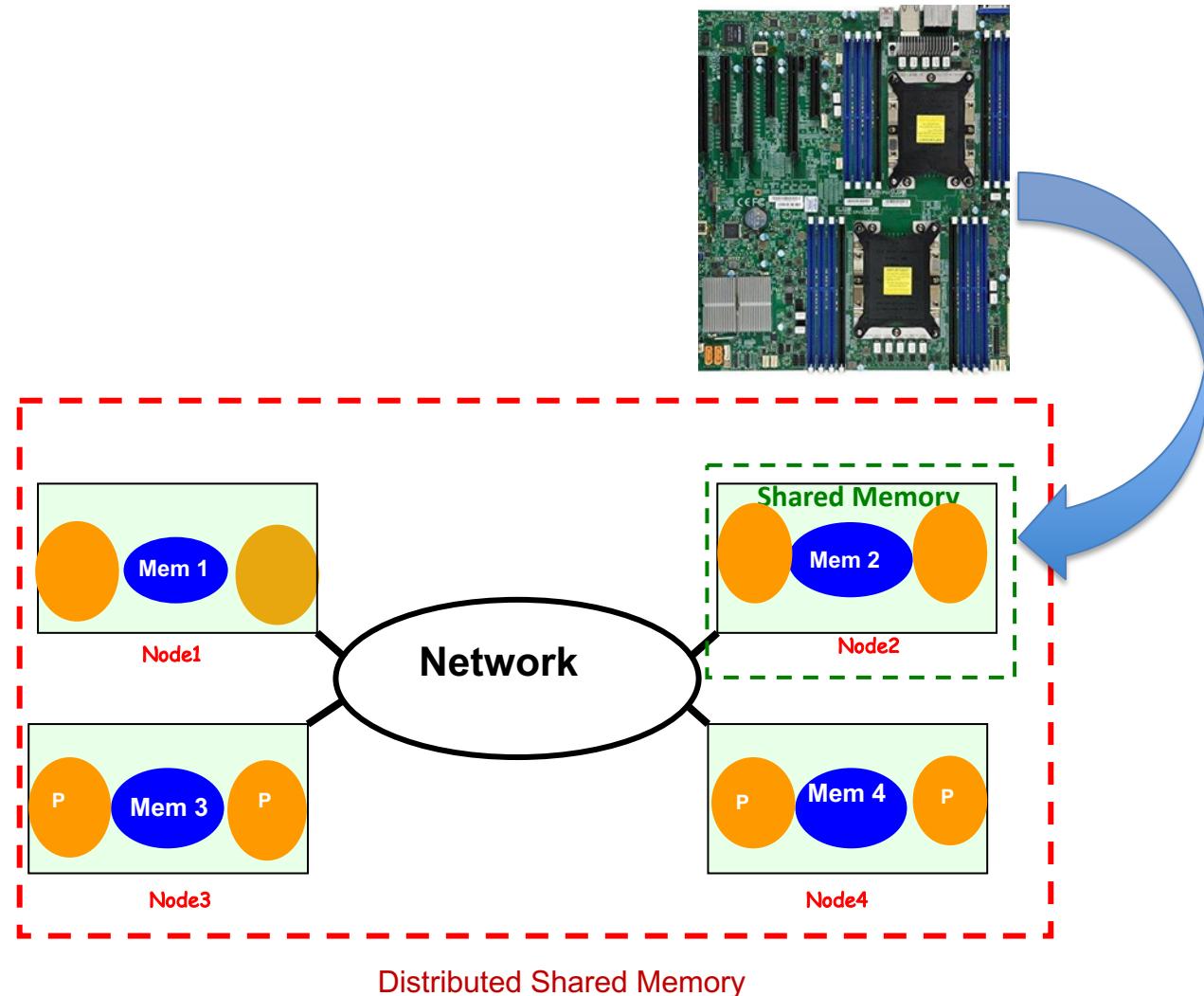


Rack



Compute Node

Distributed vs Shared Memory Systems



THE UNIVERSITY OF
CHICAGO

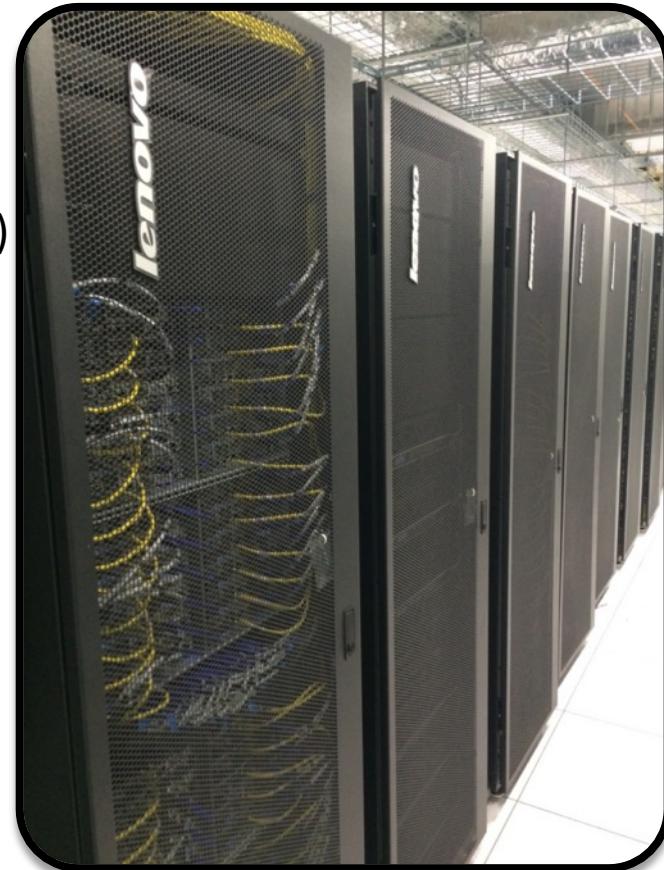
Office of Research and
National Laboratories
Research Computing Center

The RCC compute cluster

Computing hardware

367 nodes total

- **342** tightly coupled Broadwell nodes (10,360 cores)
 - Two intel E5-2680v4 processors per node (14 core/proc)
 - 155 nodes have EDR network card
 - 187 nodes have FDR network card
- **6** NVidia Tesla K80 GPU nodes (4 GPU cards/node)
- **5** large shared memory nodes (512GB each)
- **14** dual socket loosely-coupled Broadwell nodes



**RCC Manages 1515 nodes
(35,904 cores)**



THE UNIVERSITY OF
CHICAGO

Office of Research and
National Laboratories
Research Computing Center

RCC Hardware: Midway2 (Shared)

Cores(Nodes)	CPU	Memory	Network
5236 (187)	2.4GHz Intel Broadwell	64GB	EDR 100Gbit/s InfiniBand
4,340 (155)	2.4GHz Intel Broadwell	64GB	FDR14 56Gbit/s InfiniBand
140 (5)	2.4GHz Intel Broadwell	512GB	FDR14 56Gbit/s InfiniBand
504 (14)	2.4GHz Intel Broadwell	64GB	1 GigE

Cores (Nodes)	CPU Accelerator	Memory	Network
168 (6)	2.4GHz Intel Broadwell NVIDIA Tesla K80	128GB	FDR14 56bit/s InfiniBand



RCC Hardware: Midway2 (Shared)

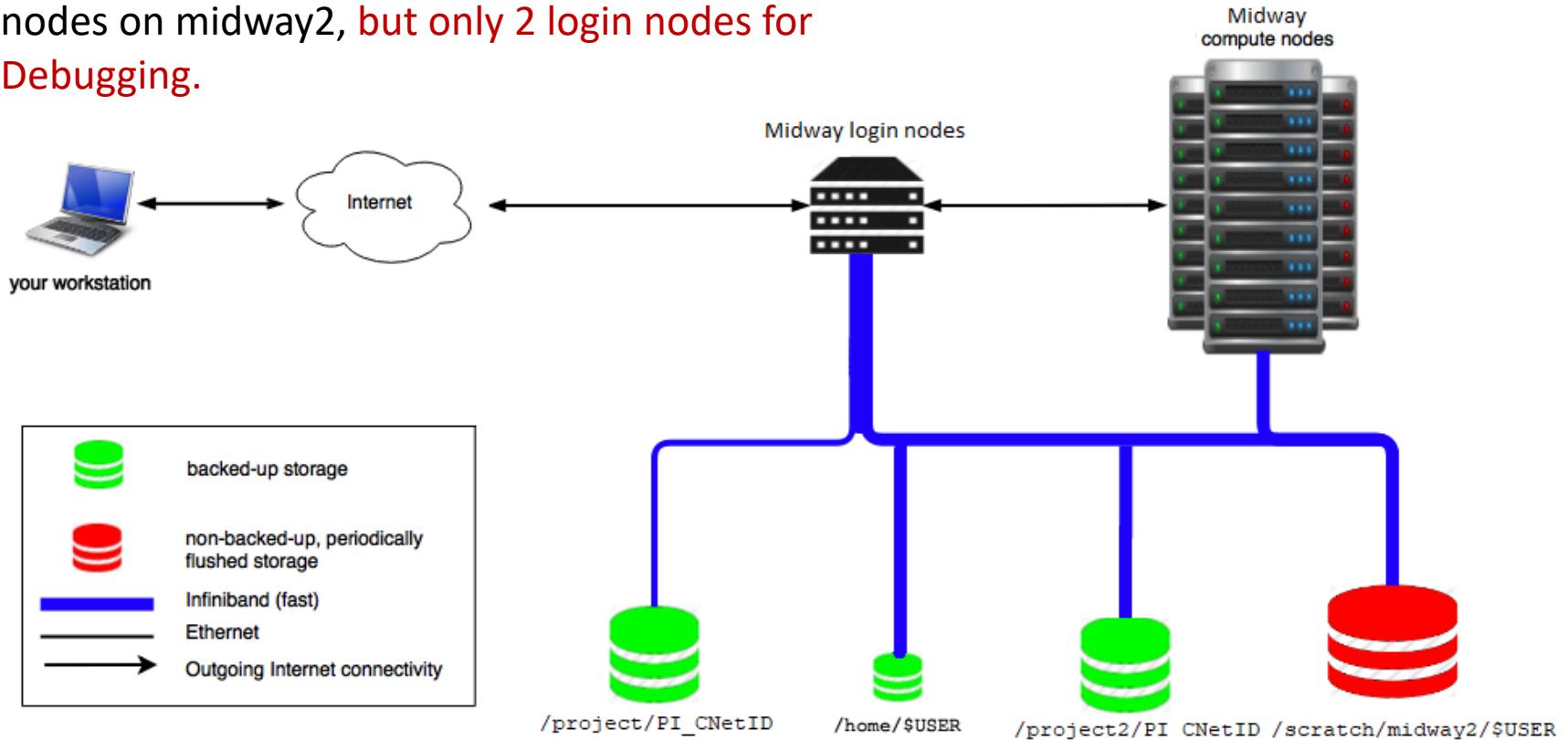
Cores(Nodes)	CPU	Memory	Slurm Queue
5180 (185)	2.4GHz Intel Broadwell	64GB	broadwl
5180 (185)	2.4GHz Intel Broadwell	64GB	broadwl
168 (6)	2.4GHz Intel Broadwell	512GB	bigmem2
504 (18)	2.4GHz Intel Broadwell	64GB	broadwl-lc

Cores (Nodes)	CPU Accelerator	Memory	Queue
168 (6)	2.4GHz Intel Broadwell NVIDIA Tesla K80	128GB	gpu2



Schematic of the Midway Cluster

There are about more than 1300+ nodes compute nodes on midway2, **but only 2 login nodes for Debugging.**



Midway2 Storage

High Capacity storage: /project2

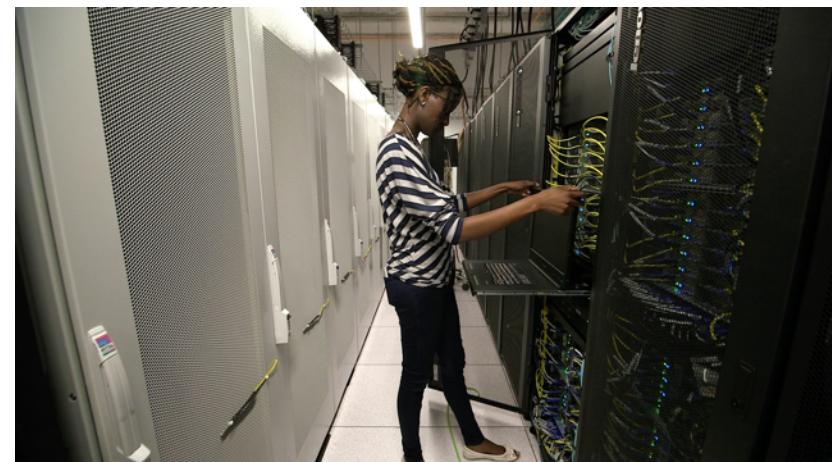
- **3.8 PB** of storage
- Backed up to tape system
- **7 daily and 4 weekly snapshots** located at /snapshots/project2
- 7 day grace period on over quota

High Performance storage: /scratch/midway2

- **190 TB** usable
- Not backed up
- 100 GB user soft quota
- 30 day grace period on over quota

Home directory space: /home

- **61 TB** of capacity
- Each user has 30 GB quota
- 7 day grace period on over quota
- **7 daily and 2 weekly snapshots** located at /snapshots/home



THE UNIVERSITY OF
CHICAGO

Office of Research and
National Laboratories
Research Computing Center

How to Run jobs at RCC

**Either Interactively or submitting
jobs to a queue using Slurm**

Repository

- Please clone the repo at
 - https://github.com/rcc-uchicago/SLURM_WORKSHOP.git

A Key point to remember

There are about more than 1300+ nodes compute nodes on midway2, but only 2 login nodes.

This means you are sharing the login nodes with many other users at once. Running intensive programs on the login nodes causes the login nodes to be slow for all other users.

- login nodes are for editing files, compiling, moving files, changing permissions, and other non-intensive tasks.
- We recommend to use *sinteractive* for interactive runs
- For long running jobs => submit them to the queue

Running Interactive jobs

- login directly to a node
 - Login to midway2.rcc.uchicago.edu
 - Run the job at the command prompt
- Run interactively using sinteractive
 - Uses Slurm to provide access to dedicated node(s) to which you can login directly
 - To use sinteractive:

```
sinteractive --time=01:00:00 --nodes=1  
--ntasks=2 --mem-per-cpu=1000  
--partition=broadwl --reservation=slurm-workshop
```



Summary of partitions on Midway2

sinfo -s

PARTITION	AVAIL	TIMELIMIT	NODES(A/I/O/T)	NODELIST
cron	up	infinite	0/4/0/4	dali-login[1-2],midway-login[1-2]
westmere	up	infinite	0/28/1/29	midway[002-030]
sandyb	up	infinite	49/24/2/75	midway[044-069-073,089,109,112,193-197,216-226,398-414,417-445,448-451]
mfj	up	infinite	1/11/0/12	midway2-[0489-0500]
test	up	infinite	0/1/0/1	midway397
cobey	up	infinite	16/0/1/17	midway2-[0217-0220,0409-0410,0427-0436],midway2-bigmem05
jnovembre	up	infinite	4/4/0/8	midway2-[0401-0408]
tas1	up	infinite	0/19/5/24	midway[783-806]
gpu	up	infinite	0/8/1/9	midway[230-232,493-494],midway-l34-[01-04]
viz	up	infinite	0/1/0/1	midway229
mic	up	infinite	0/0/2/2	midway-mic[01-02]
sepalmer	up	infinite	0/3/0/3	midway[453-454,590]
kicp	up	infinite	0/20/11/31	midway[159-188,191]
kicp-long	up	infinite	0/20/11/31	midway[159-188,191]
kicp-ht	up	infinite	0/4/0/4	midway[151-152,189-190]
surph	up	infinite	0/7/7/14	midway[143-150,153-158]
surph-large	up	infinite	0/27/18/45	midway[143-150,153-188,191]
xenon1t	up	infinite	14/2/0/16	midway2-[0411-0426]

⇒ NODES(A/I/O/T) : Nodes (Allocated/Idle/Other/Total)

e.g: To check the Broadwl partition: `sinfo -s |grep broadwl`

Slurm: Some key terms to remember

A **job** is the resources you are using and the code you are running

The **queue** in Slurm is all RUNNING and all PENDING jobs
To see every job in the queue on Oscar, use the command

```
squeue
```

To see your jobs in the queue

```
squeue -u <cnetid>
```

```
or
```

```
myq
```

What goes into a batch script?

A batch script is list of instructions for slurm.

```
#!/bin/bash

# Here is a comment
#SBATCH --time=1:00:00

#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=2000
#SBATCH --job-name=MyJob
#SBATCH --output= MyJob-%j.out
#SBATCH --error=MyJob-%j.err

module load <module name>
#Run your code
```

What goes into a batch script?

A batch script is list of instructions for slurm.

```
#!/bin/bash

# Here is a comment
#SBATCH --time=1:00:00

#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=2000
#SBATCH --job-name=MyJob
#SBATCH --output= MyJob-%j.out
#SBATCH --error=MyJob-%j.err

module load <module name>
#Run your code
```

This **#!** is a shebang

It tells operating system to use
/bin/bash
with this script

What goes into a batch script?

A batch script is list of instructions for slurm.

```
#!/bin/bash

# Here is a comment
#SBATCH --time=1:00:00

#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=2000
#SBATCH --job-name=MyJob
#SBATCH --output= MyJob-%j.out
#SBATCH --error=MyJob-%j.err

module load <module name>
#Run your code
```

is a comment
everything after # is ignored by
bash

What goes into a batch script?

A batch script is list of instructions for slurm.

```
#!/bin/bash

# Here is a comment
#SBATCH --time=1:00:00

#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=2000
#SBATCH --job-name=MyJob
#SBATCH --output= MyJob-%j.out
#SBATCH --error=MyJob-%j.err

module load <module name>
#Run your code
```

#SBATCH is a directive

It is a comment in Bash

#SBATCH is only relevant to slurm:
sbatch my_script.sh

What goes into a batch script?

A batch script is list of instructions for slurm.

```
#!/bin/bash

# Here is a comment
#SBATCH --time=1:00:00

#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=2000
#SBATCH --job-name=MyJob
#SBATCH --output= MyJob-%j.out
#SBATCH --error=MyJob-%j.err

module load <module name>
#Run your code
```

#SBATCH is a directive

It is a comment in Bash

#SBATCH is only relevant to slurm:
sbatch my_script.sh

To comment out directives, break
the pattern, e.g.

```
##SBATCH
# SBATCH
```

What goes into a batch script?

A batch script is list of instructions for slurm.

```
#!/bin/bash

# Here is a comment
#SBATCH --time=1:00:00

#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=2000
#SBATCH --job-name=MyJob
#SBATCH --output= MyJob-%j.out
#SBATCH --error=MyJob-%j.err

module load <module name>
#Run your code
```

Instructions for Slurm must go at the top of the script

Any #SBATCH lines you put after your program will be ignored

What goes into a batch script?

A batch script is list of instructions for slurm.

```
#!/bin/bash

# Here is a comment
#SBATCH --time=1:00:00

#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=2000
#SBATCH --job-name=MyJob
#SBATCH --output= MyJob-%j.out
#SBATCH --error=MyJob-%j.err

module load <module name>
#Run your code
```

Slurm has some variables you can use. %j is the job number. When the job runs %j will be expanded to the job number. In this example %j is used in the output file and error file names:

MyJob-**13571056**.out
MyJob-**13571056**.err

%j is unique. By using %j in your filenames you guarantee a unique file name, which means you won't accidentally overwrite previous output.

What goes into a batch script?

A batch script is list of instructions for slurm.

```
#!/bin/bash
```

```
# Here is a comment
```

```
#SBATCH --time=1:00:00
```

=> Time your job is allowed to run

```
#SBATCH --nodes=1
```

=> Number of nodes to run on

```
#SBATCH --ntasks-per-node=1
```

=> Number of cores on each node to use

```
#SBATCH --mem-per-cpu=2000
```

=> Memory per cpu => 2000Mb or 2Gb

```
#SBATCH --job-name=MyJob
```

=> Name of the job.

```
#SBATCH --output= MyJob-%j.out
```

=> Job output file behaves as stdout for the code.

```
#SBATCH --error=MyJob-%j.err
```

=> Error file. behaves as stderr for the code.

```
module load <module name>
```

=> Load any modules you need for your application

```
#Run your code
```

=> run the code you want

Running batch jobs using a Submission Script

- A simple job submission script (saved as python.sbatch):

```
#!/bin/bash
#SBATCH --job-name=first_python_job
#SBATCH --output=first_python_job_%j.out
#SBATCH --error=first_python_job_%j.err
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=2000M
#SBATCH --partition=broadwl
#SBATCH --reservation=kicpworkshop-cpu
#SBATCH --time=00:30:00
module load python
python hello_world.py
echo "job finished at `date`"
```

Slurm

Your Job

- To submit the above script:
 - **sbatch** python.sbatch



Please do the Exercise;
Ex-1 in Repo

How to submit OpenMP jobs?

```
#!/bin/bash

#Here is a comment
#SBATCH --time=1:00:00
#SBATCH --partition=broadwl
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --mem-per-cpu=2000
#SBATCH --job-name=MyJob
#SBATCH --output= MyJob-%j.out
#SBATCH --error=MyJob-%j.err

module load <module name>
export OMP_NUM_THREADS=8
#Run your code
./my_executable
```

Specify number of cores > 1.

OMP_NUM_THREADS is an environment variable.

Please do the Exercise;
Ex-2 in Repo

How to submit Parallel MPI jobs?

```
#!/bin/bash

#Here is a comment
#SBATCH --time=1:00:00
#SBATCH -job-name=MyJob
#SBATCH -output= MyJob-%j.out
#SBATCH -error=MyJob-%j.err
#SBATCH -partition=broadwl
#SBATCH -nodes=4
#SBATCH -ntasks-per-node=8
#SBATCH --mem-per-cpu=2000
module load openmpi
module load <module name>
#Run your code
mpirun ./my_executable
```

Specify number of nodes > 1.

Specify number of cores >= 1.

Load OPENMPI MPI library or IntelMPI

Please do the Exercise;
Ex-3 in Repo

How to submit GPU jobs?

```
#!/bin/bash

#Here is a comment
#SBATCH --time=1:00:00
#SBATCH --partition=gpu2
#SBATCH --gres=gpu:1
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --mem-per-cpu=2000
#SBATCH --job-name=MyJob
#SBATCH --output= MyJob-%j.out
#SBATCH --error=MyJob-%j.err

module load <module name>
#Run your code
./my_executable
```

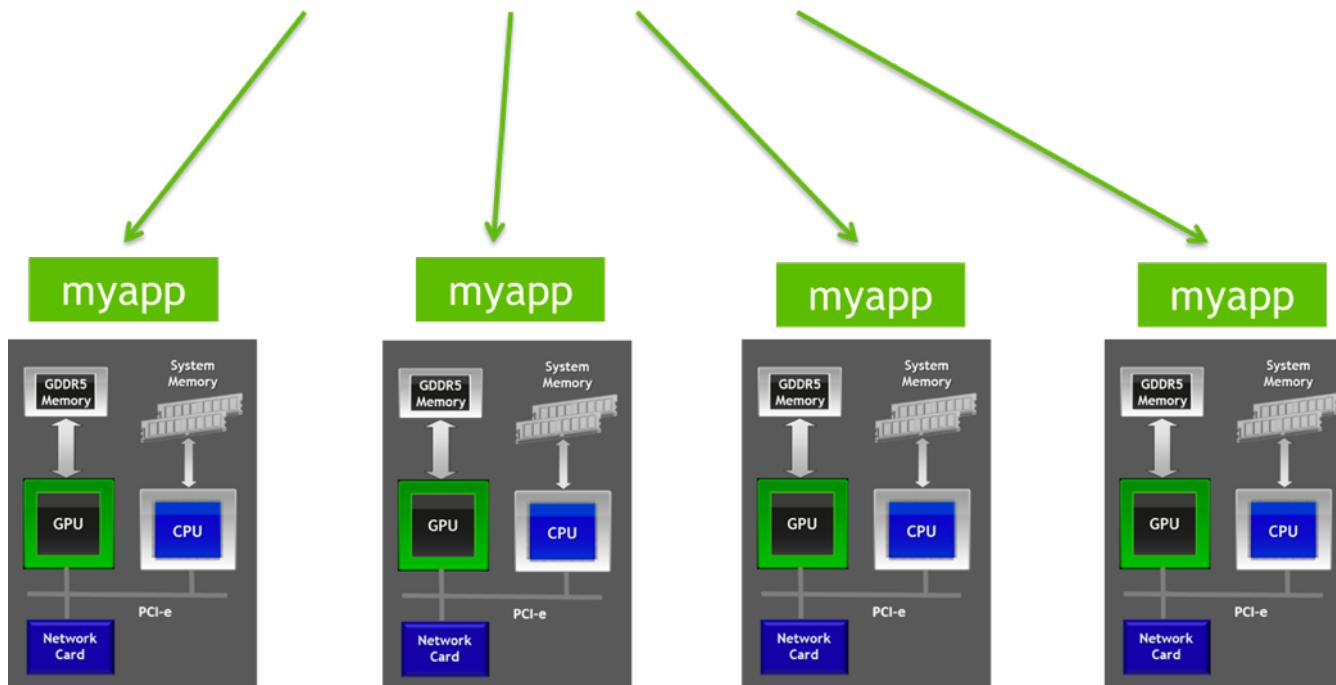
Specify partition gpu2
Specify number of gpus, like *gpu:1*

Please do the Exercise;
Ex-4 in Repo

How to submit MPI + GPU jobs?

CUDA Aware MPI

```
mpirun -np 4 ./myapp <args>
```



How to submit MPI + GPU jobs?

Compilation

```
#!/bin/bash .brown.edu/oscar/gpu-computing/mpi-cuda
SUBMITTING JOBS

module load openmpi/3.1.2
module load cuda/10.1
# Compiling the device code
nvcc -c dev.cu
#Compiling the host code
mpicc -c hostname.c

# Linking the host and device code
mpicc -o HostMap dev.o hostname.o -lcudart

# Submitting the job as batch script
sbatch mpijob.sh
```

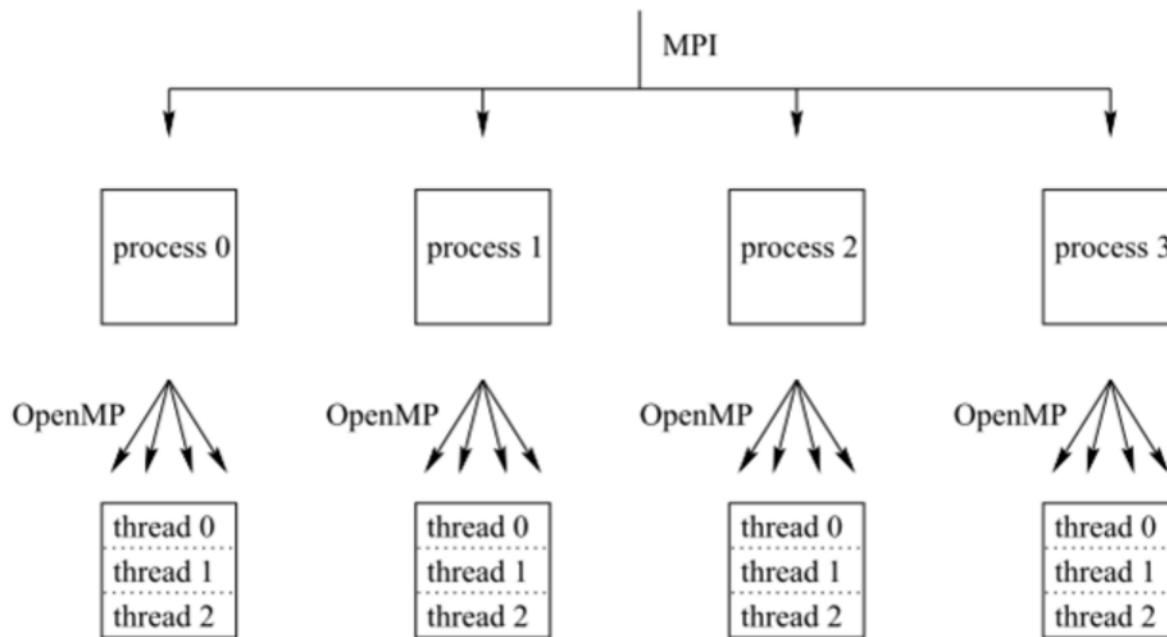
Job Submission

```
#!/bin/bash .brown.edu/oscar/gpu-computing/mpi-cuda
SUBMITTING JOBS

#SBATCH -t 00:30:00
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH --partition=gpu2
#SBATCH --gres=gpu:2
#SBATCH --job-name=MyJob
#SBATCH --output=MyJob-%j.out
#SBATCH --error=MyJob-%j.err
#SBATCH --qos=stafftest
mpirun ./HostMap
```

Please do the Exercise;
Ex-5 in Repo

How to submit MPI + OpenMPI jobs?



```
#!/bin/bash
#SBATCH --job-name=hybrid
#SBATCH --output=hybrid_%j.out
#SBATCH --error=hybrid_%j.err
#SBATCH --time=00:10:00
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=8
#SBATCH --partition=broadwl
#SBATCH --constraint=edr
#SBATCH --qos=stafftest

# Load the default OpenMPI module.
module load openmpi

# Set OMP_NUM_THREADS to the number of CPUs per task we asked for.
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

mpirun ./mpomp
```

Please do the Exercise;
Ex-5a in Repo

How to submit array based jobs

```
#!/bin/bash
# Job Name
#SBATCH --job-name=arrayjob
# Walltime requested
#SBATCH --time=0:10:00
#Add partition
#SBATCH --partition=broadwl-lc

# Provide index values (TASK IDs)
#SBATCH --array=1-16

# Use '%A' for array-job ID, '%J' for job ID and '%a' for task ID
#SBATCH --error=maths%A-%a.err
#SBATCH --output=maths%A-%a.out

# single core
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=2000

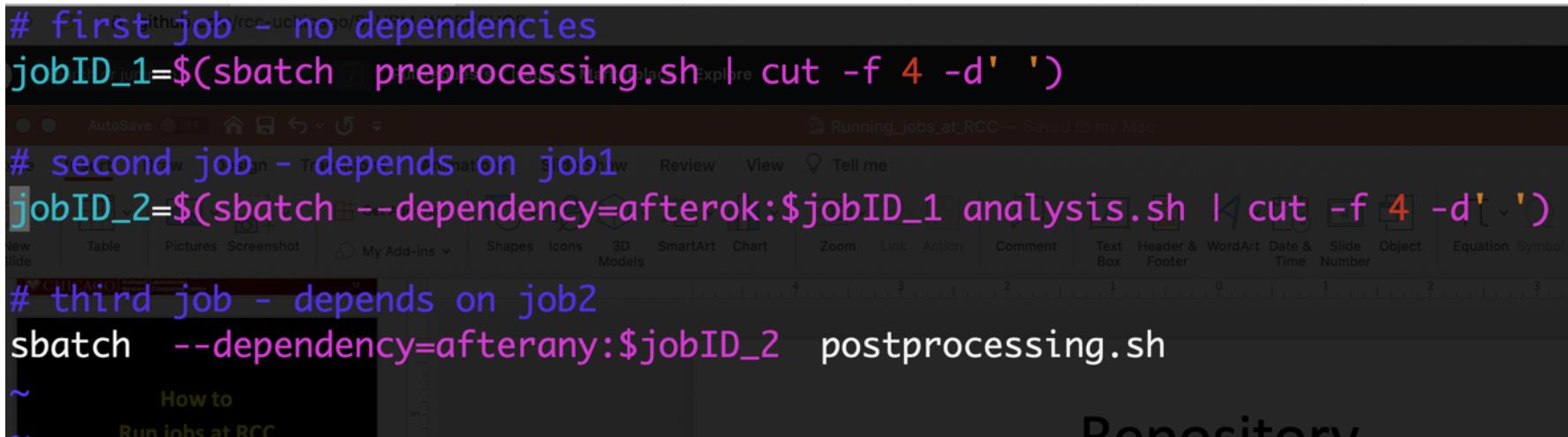
# Use the $SLURM_ARRAY_TASK_ID variable to provide different inputs for each job
input=$((SLURM_ARRAY_TASK_ID*1000+2))
echo "Running job array number: "$SLURM_ARRAY_TASK_ID "input " $input
```

Please do the Exercise;
Ex-6 in Repo

How to submit dependent jobs

SLURM Rule:

sbatch --dependency=type:job_id jobfile



A screenshot of a Microsoft Word document titled "Running_jobs_at_RCC". The document contains the following text:

```
# first job - no dependencies
jobID_1=$(sbatch preprocessing.sh | cut -f 4 -d' ')

# second job - depends on job1
jobID_2=$(sbatch --dependency=afterok:$jobID_1 analysis.sh | cut -f 4 -d' ')

# third job - depends on job2
sbatch --dependency=afterany:$jobID_2 postprocessing.sh
```

The code demonstrates how to submit multiple SLURM jobs. The first job has no dependencies. The second job depends on the completion of the first job (afterok). The third job depends on either the completion or cancellation of the second job (afterany).

How to submit dependent jobs

after	This job can begin execution after the specified jobs have begun execution
afterany	This job can begin execution after the specified jobs have terminated.
aftercorr	A task of this job array can begin execution after the corresponding task ID in the specified job has completed successfully
afternotok	This job can begin execution after the specified jobs have terminated in some failed state
afterok	This job can begin execution after the specified jobs have successfully executed
singleton	This job can begin execution after any previously launched jobs sharing the same job name and user have terminated

Please do the Exercise;
Ex-7 in Repo

How to submit Parallel batch jobs

```
#!/bin/sh

#SBATCH --time=01:00:00
#SBATCH --partition=broadwl
#SBATCH --ntasks=28
#SBATCH --mem-per-cpu=2G # NOTE DO NOT USE THE --mem= OPTION
# Load the default version of GNU parallel. module load parallel
# set the max number of processes (which determine the max per processor)
ulimit -u 10000
# This specifies the options used to run srun. The "-N1 -n1" options are
# used to allocates a single core to each task.
srun="srun --exclusive -N1 -n1"
#Run GNU parallel
parallel="parallel --delay 0.2 -j $SLURM_NTASKS --joblog runtask.log --resume"
# Run a script, runtask.sh, using GNU parallel and srun.
$parallel "$srun ./runtask.sh arg1:{1} > runtask.sh.{1}" ::: {1..128}
# Note that if your program does not take any input, use the -n0 option to call the
parallel command: ## $parallel -n0
"$srun ./run_noinput_task.sh > output.{1}" ::: {1..128}
```

How to submit Parallel batch jobs

Please do the Exercise;
Ex-8 in Repo

What resources should I ask for?

This depends on the code you are running

What resources should I ask for?

This depends on the code you are running

What resources should I ask for?

This depends on the code you are running

Nodes/Cores

- Question: is your code parallel? You will need to find out if your code can
 - Run on multiple cores? Run across multiple nodes?
 - Check if your code is threaded, multiprocessor, MPI
- Question: Is your code serial?
 - This means it can only make use of one core

What resources should I ask for?

This depends on the code you are running

Wall Time

Make an estimate of your job run and add a bit.

e.g. if think your code will take an hour, give it 1 hour and 30 min

- If your job runs out of time, your job will be killed, so
- be accurate with your estimate without going below.

What resources should I ask for?

This depends on the code you are running

Memory

For memory, this can take some trial and error. You can ask for a lot, then measure your usage. If you have asked for more memory and then reduce your memory with the next job.

- To ask for all the memory available on a node, use `#SBATCH --mem=0`

What if I need an entire node or specific features?

Add this in your batch script

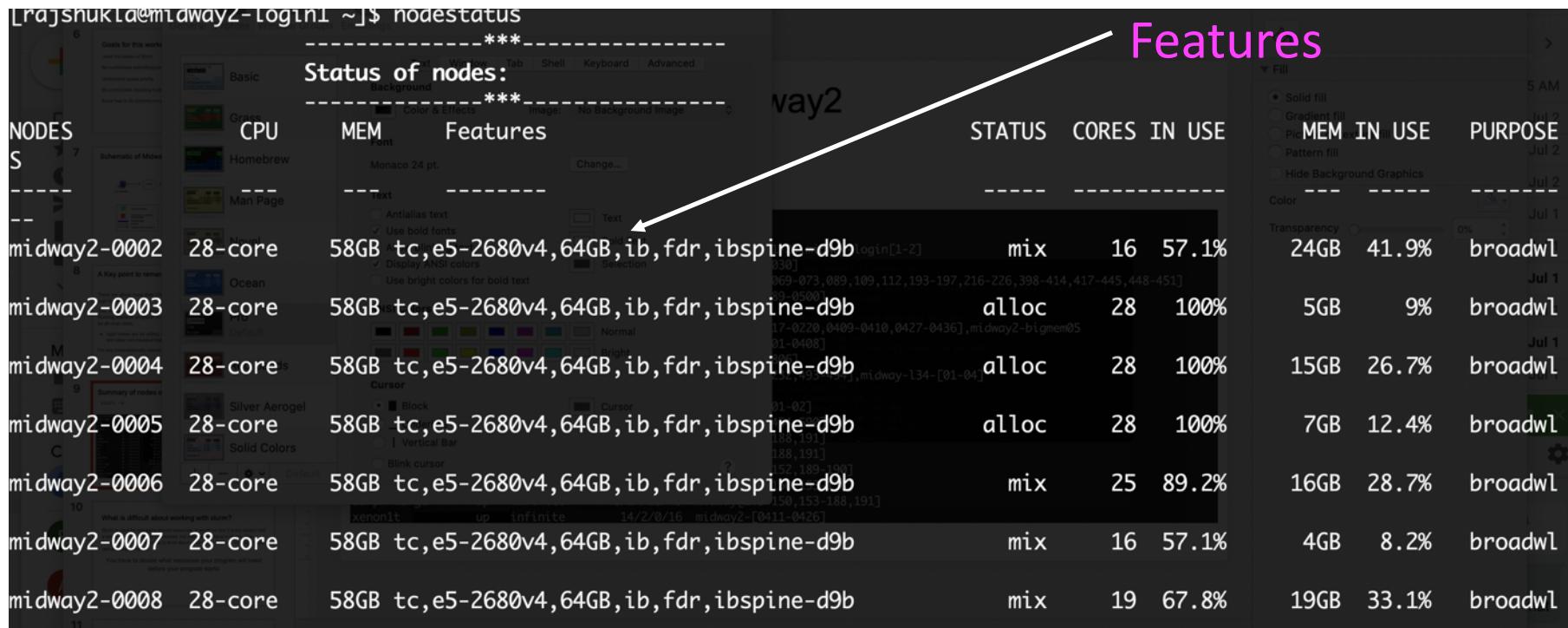
```
#SBATCH –exclusive
```

Add this in your batch script

```
#SBATCH –constraint=v100
```

How do I know the features of the node to use with #SBATCH -constraint?

nodestatus



The screenshot shows the terminal output of the `nodestatus` command, which lists nodes and their configurations. A pink arrow points from the word "Features" in the terminal output to the "Features" section of a configuration interface overlaid on the terminal window.

NODES	NAME	TYPE	MEMORY	FEATURES	STATUS	CORES IN USE	MEM IN USE	PURPOSE
midway2-0002	midway2-0002	28-core	58GB	tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b	mix	16	57.1%	broadwl
midway2-0003	midway2-0003	28-core	58GB	tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b	alloc	28	100%	broadwl
midway2-0004	midway2-0004	28-core	58GB	tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b	alloc	28	100%	broadwl
midway2-0005	midway2-0005	28-core	58GB	tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b	alloc	28	100%	broadwl
midway2-0006	midway2-0006	28-core	58GB	tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b	mix	25	89.2%	broadwl
midway2-0007	midway2-0007	28-core	58GB	tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b	mix	16	57.1%	broadwl
midway2-0008	midway2-0008	28-core	58GB	tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b	mix	19	67.8%	broadwl

What resources should I ask for?

This depends on the code you are running

GPUs If your code is built to use GPUs you can submit to the gpu partition. To request 1 GPU:

```
#SBATCH -p gpu2 --gres=gpu:1
```

What resources did my job actually use?

It is good practice to occasionally check what resources your job is using. For example if you are going to be submitting hundreds of similar jobs, you may save yourself a lot of waiting time in the queue by checking that you are not over requesting resources. Midway2 has a script to display the resources a job used:

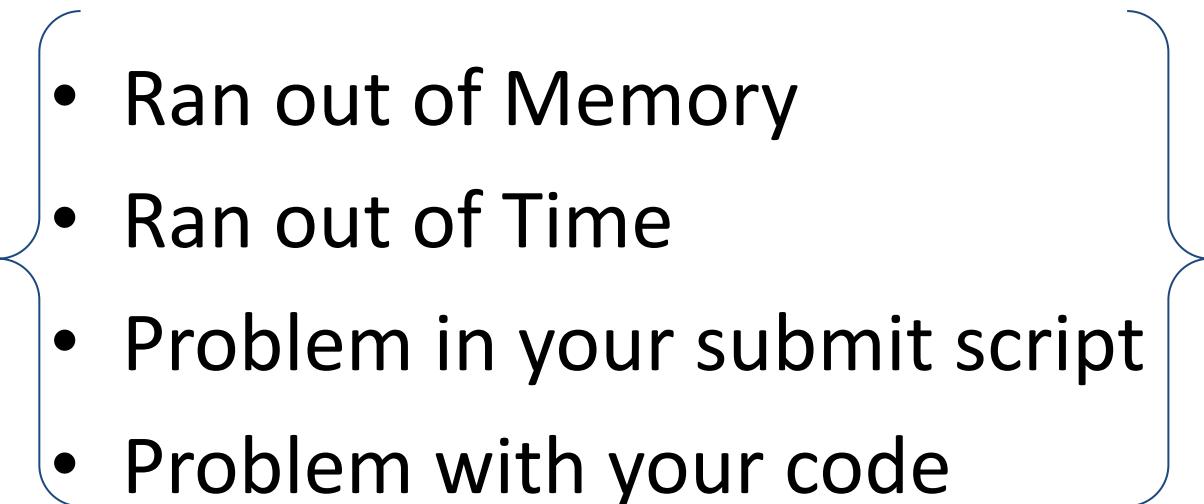
```
jobinfo -j 99999
```

Replace 99999 with the job ID of the job you are interested in.

Why did my job fail?

- Ran out of Memory
- Ran out of Time
- Problem in your submit script
- Problem with your code
- Node failure

Why did my job fail?

- Ran out of Memory
 - Ran out of Time
 - Problem in your submit script
 - Problem with your code
 - Node failure
- 
- You can fix these

Why did my job fail?

- Ran out of Memory
- Ran out of Time
- Problem in your submit script
- Problem with your code
- Node failure

You can fix these

=> For node failure – please email
help@rcc.uchicago.edu, if this happens

Job Priority

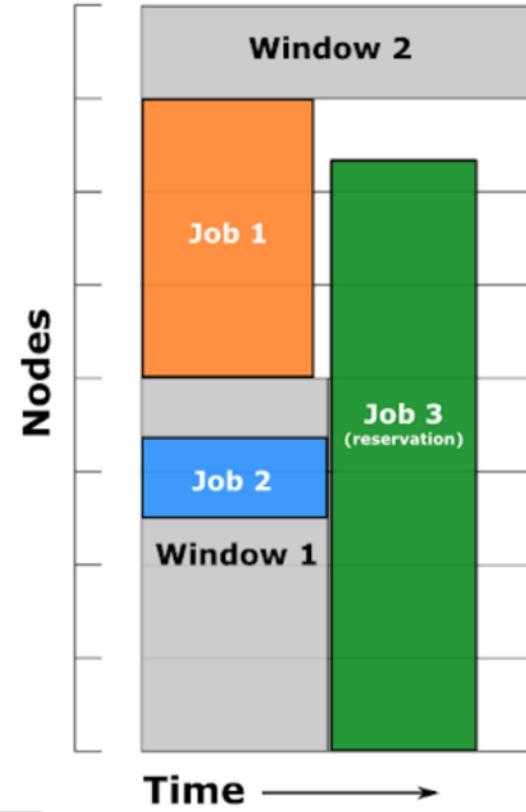
- Priority is calculated using a **Fairshare** algorithm
- Fairshare is function of
 - Requested wall clock, memory, nodes/cores, etc.
 - Length of time in queue
 - Number of jobs in a time window and per PI group
 - Backfill
 - Etc.

Job Priority

Backfill



This blue line
represents all the
cores on Midway



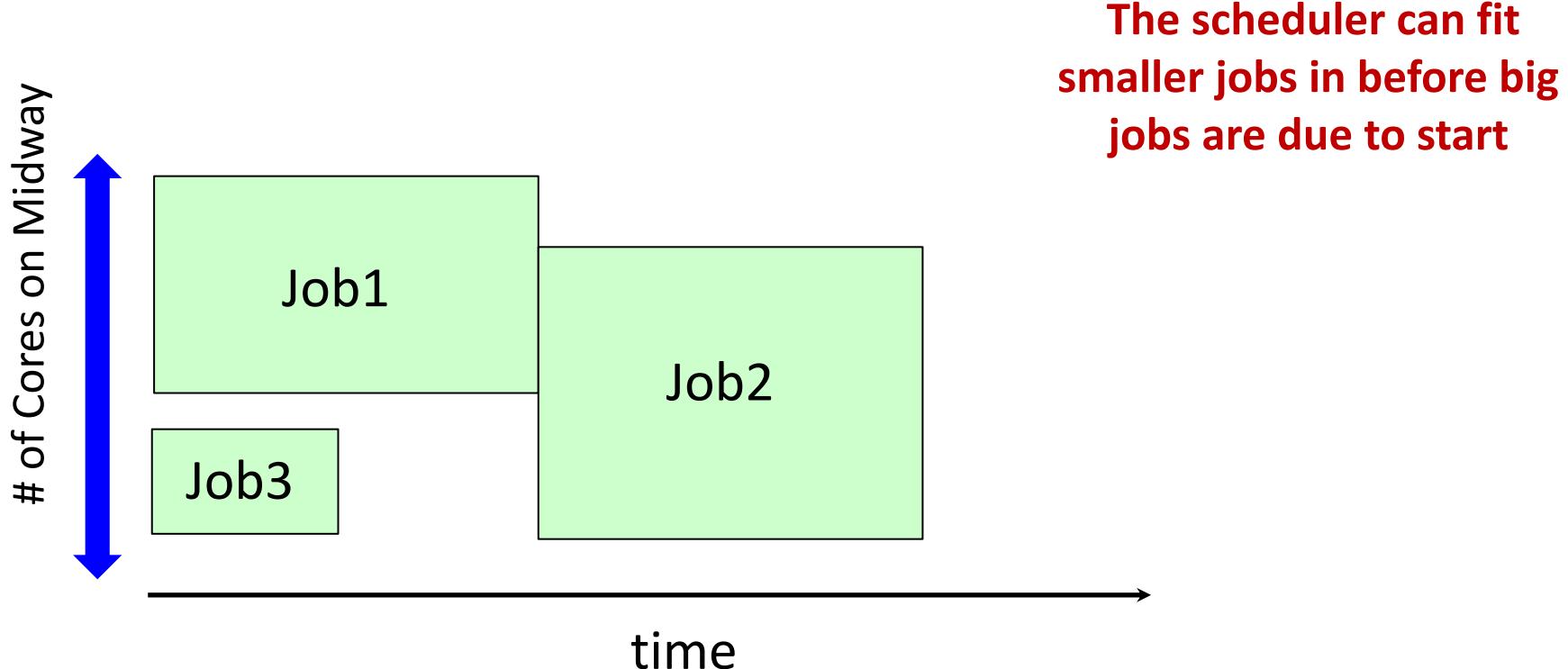
time

→ the x axis is time

Job Priority

Backfill

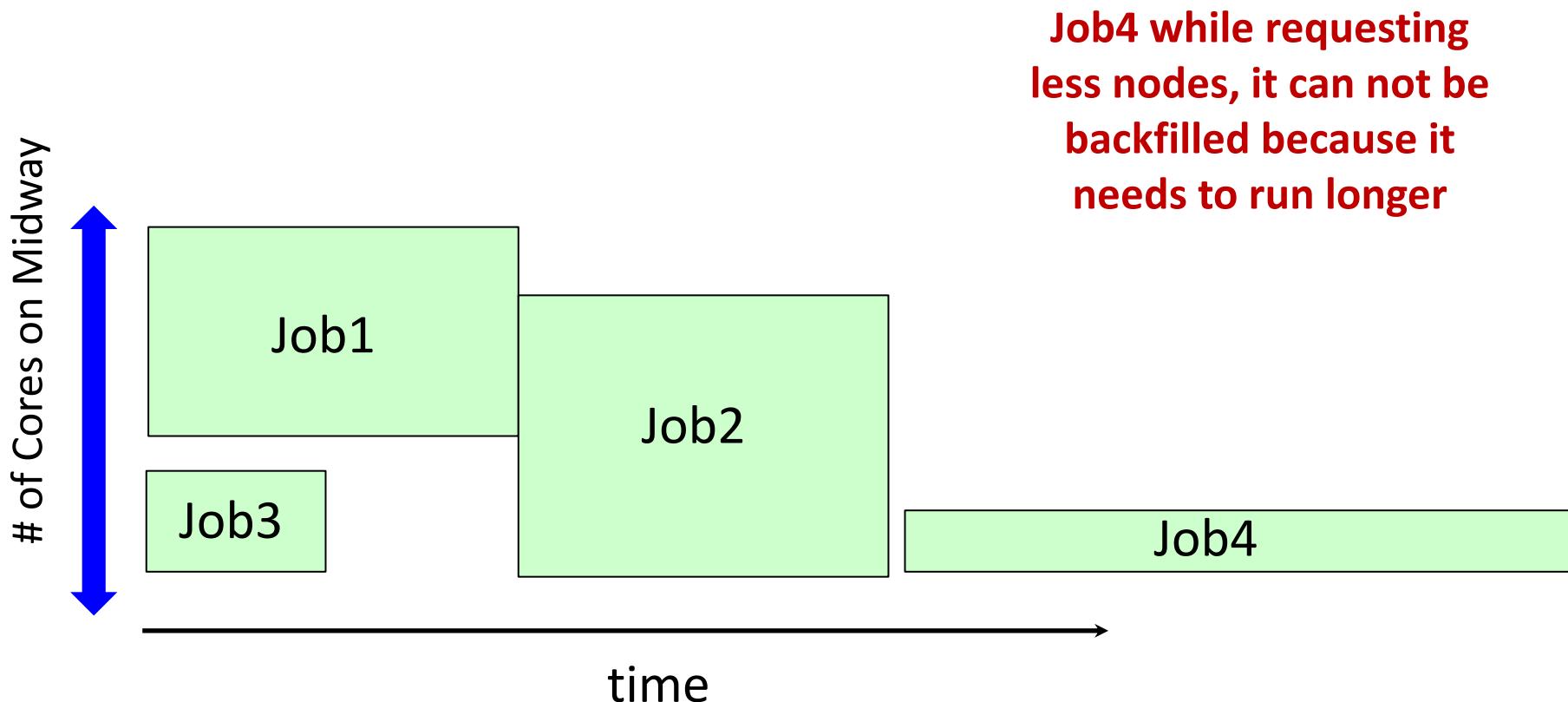
Why has someone else's job started before mine?



Job Priority

Backfill

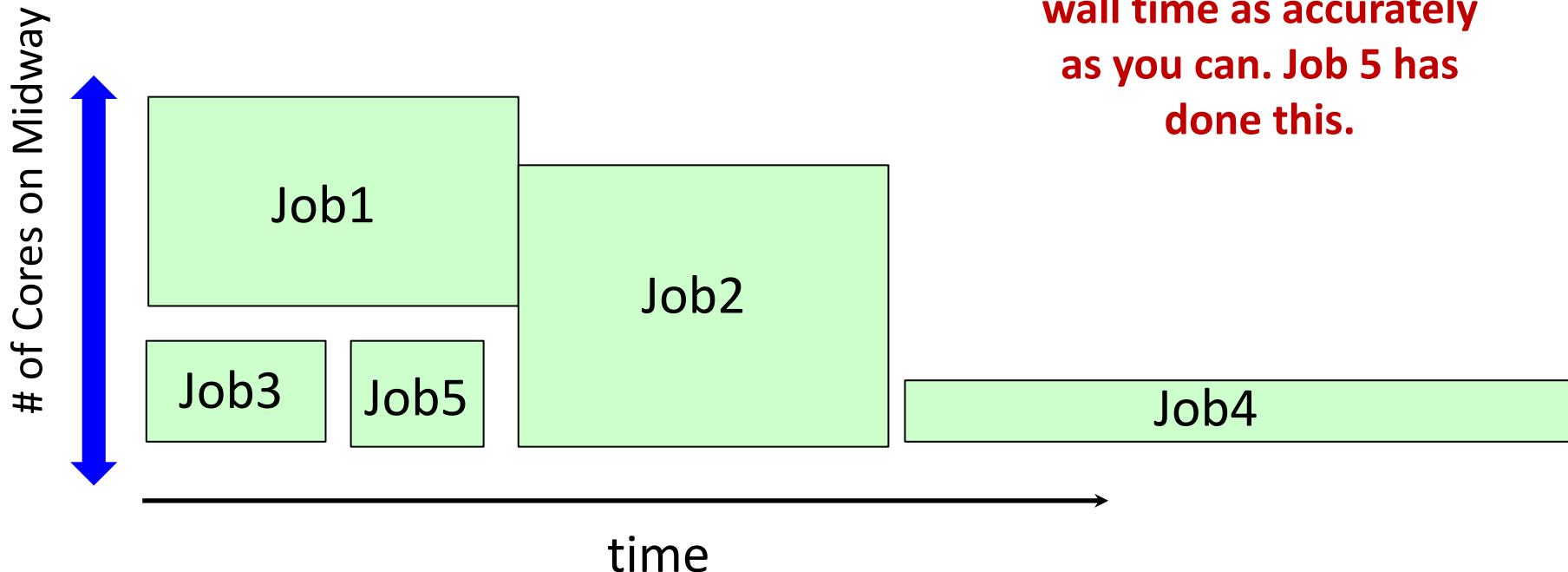
Why has someone else's job started before mine?



Job Priority

Backfill

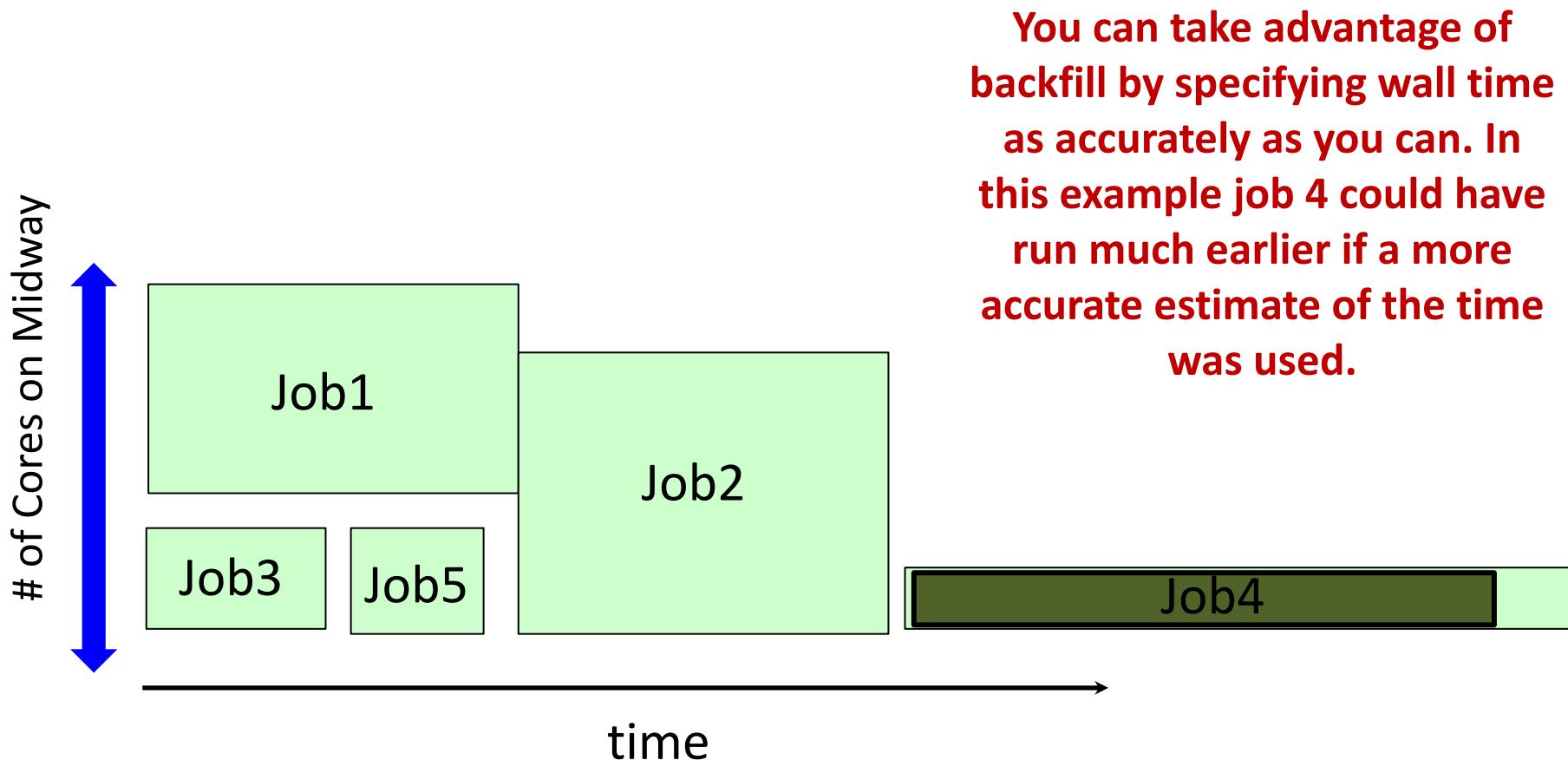
Why has someone else's job started before mine?



Job Priority

Backfill

Why has someone else's job started before mine?



Job submission and monitoring

SLURM Commands

Command	Description
<code>sbatch script.sbatch</code>	Submits <code>script.sbatch</code> job script
<code>squeue -u \$USER or myq</code>	Reports the status of your jobs
<code>sacct -u \$USER</code>	Displays accounting data for your job(s)
<code>scancel jobid</code>	Cancels a running job or removes it from the queue
<code>scontrol show job jobid or jobinfo</code>	Displays details of a running job



THE UNIVERSITY OF
CHICAGO

Office of Research and
National Laboratories
Research Computing Center

Recommended online resources

- User guide on running jobs on Midway
 - <https://rcc.uchicago.edu/docs/running-jobs/index.html>
- Details Slurm documentation
 - <https://slurm.schedmd.com/sbatch.html>
- SLURM Cheat Sheet
 - <https://slurm.schedmd.com/pdfs/summary.pdf>

RCC Help

A bottom-up
approach to
supporting
research
computing

Contact:

- By email: help@rcc.uchicago.edu
- Web: rcc.uchicago.edu
- Phone: 773-795-2667
- In person:
 - 5607 S Drexel Avenue
 - Regenstein – Room 216
- Workshops and Tutorials:
 - <http://rcc.uchicago.edu/services/training.html>

Thank You!