THE UNIVERSITY OF CHICAGO | Research Computing Center

Slurm Resource Scheduler and Other Best Practices for Using the Computing Cluster

Jonathan Skone

Today's Agenda

SLURM Job Submission on the HPC Cluster

Objectives:

- Be comfortable submitting jobs on the RCC HPC Cluster
- Understand queue priority
- Ability to query status of job and understand exit codes
- Know how to fix common errors
- Using array jobs and GNU parallel

Using DASK on RCC resources (led by Ivan)

Connecting to Miday2 and Setup

Midway2 logins: **midway2.rcc.uchicago.edu** Midway 2

- **Mac/Linux users/Windows Powershell users:**
  Open a terminal app and issue the following command.

```
w/o X11 [johnnyb@volpe]$ ssh $USER@midway2.rcc.uchicago.edu
```

Please see the following link for additional instructions on setting up your client and connecting to Midway2 resources.

Connecting to midway2 Instructions

- **Get the materials for today's workshop:**
  After logging in to midway2 login nodes. Use the copy 'cp' command to copy today's materials to your scratch directory $SCRATCH

```
[johnnyb@m21]$ cd $SCRATCH ; cp /project2/env_bootcamp/materials/jobs ./
```

Remember to Respect the Login Nodes

There are more than 1300+ compute nodes that are part of midway2, but there are only 2 login nodes that server about 300 active users.

There is a process that monitors the login nodes and will kick out users who are using the logins inappropriately.

.Please adhere to the following practices for login node use

- 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

Review of the RCC compute cluster

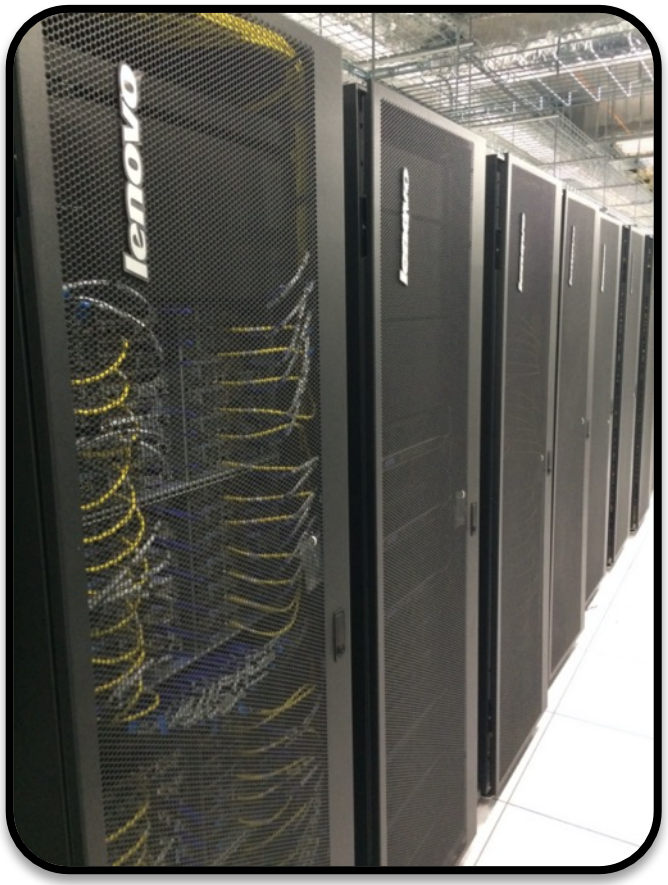
Computing hardware

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

Cluster Partnership Program: 1000+ nodes

- **900+** tightly coupled infiniband nodes
- **20+** Big memory nodes
- **100+** Nvidia GPU nodes



RCC Manages 1515 nodes (35,904 cores)

RCC Midway2 Shared Hardware

To query the list of communal resources that users can submit jobs to use the rcchelp command.

```
[johnnyb@m21]$ rcchelp sinfo 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

NOTE: For this bootcamp you only have access to the **edu** partition

RCC Job Scheduler: SLURM

Open source and very powerful
job scheduler.

Most HPC centers have migrated to using
This scheduler.

We are currently running slurm v. 18.08.8

SLURM documentation
https://slurm.schedmd.com/sbatch.html

Review: Running Jobs Interactively

- Accessing a compute node interactively will allow you to run the job at the command prompt directly on the compute node.

- To run interactively use the sinteractive command
  - Uses Slurm to provide access to dedicated node(s) to which you can login directly
  - To use sinteractive:

```
 sinteractive --time=00:30:00 --nodes=1
--ntasks=2 --mem-per-cpu=1000
--partition=edu
```

- **Note:** If you close your terminal you will NOT kill the interactive session. To exit the interactive session, type “exit” from the compute node or use **scancel**

Slurm: Some key terms to remember

When we refer to a **job** we mean a task or set of tasks that you have you have scheduled to use the compute resources through the SLURM scheduler.

The **queue** is all jobs in a particular partition. They can be in the RUNNING, PENDING, DEPENDENCY, COMPLETING, or HELD state. To see every job in a particular partitions queue, use the command squeue –p <partition>

```
[johnnyb@m21]$ squeue –p edu
```

To see only your jobs in the queue use squeue –u or myq

```
[johnnyb@m21]$ squeue –u $USER

[johnnyb@m21]$ 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.
#SBATCH –mail-user=jb@uchicago.edu      => User to mail. Email address to send job notices
#SBATCH –mail-user=jb@uchicago.edu      ⇒  Type of email notices to be sent to user.
#SBATCH --mail-type=ALL

 module load <module name>              => Load any modules you need for your application
 python myjob.py                        => run the code you want
```

Running batch jobs using a Submission Script

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

Slurm

```
#!/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
```

Your Job

```
module load python
python hello_world.py
echo "job finished at `date`"
```

- To submit the above script:
  - sbatch job.sbatch

Please do Exercise 1 in the job folder

```
[johnnyb@m21]$ cd $SCRATCH

[johnnyb@m21]$ cd jobs/Ex-1

[johnnyb@m21]$ cat job.sbatch

[johnnyb@m21]$ sbatch job.sbatch

[johnnyb@m21]$ squeue –u $USER
```

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
```

Applicable to using any python package that relies on threaded BLAS routines coming from Intel MKL

Specify number of cores > 1.

OMP_NUM_THREADS is an environment variable.

Please do Exercise 2 in the job folder

```
[johnnyb@m21]$ cd $SCRATCH

[johnnyb@m21]$ cd jobs/Ex-2

[johnnyb@m21]$ cat job.sbatch

[johnnyb@m21]$ sbatch job.sbatch

[johnnyb@m21]$ squeue –u $USER
```

Note: here we use a Makefile to manage building the program. One can create tasks that are programmatically called using a Makefile

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=edu
#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
```

Always use –ntasks-per-node for MPI jobs
In combination w/ --nodes flag

Specify number of nodes > 1.

Specify number of cores >= 1.

Load OPENMPI MPI library or IntelMPI

Please do Exercise 3 in the job folder

```
[johnnyb@m21]$ cd $SCRATCH

[johnnyb@m21]$ cd jobs/Ex-3

[johnnyb@m21]$ cat job.sbatch

[johnnyb@m21]$ sbatch job.sbatch

[johnnyb@m21]$ squeue –u $USER
```

Note: here we use a Makefile to manage building the program. One can create tasks that are programmatically called using a Makefile

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
```

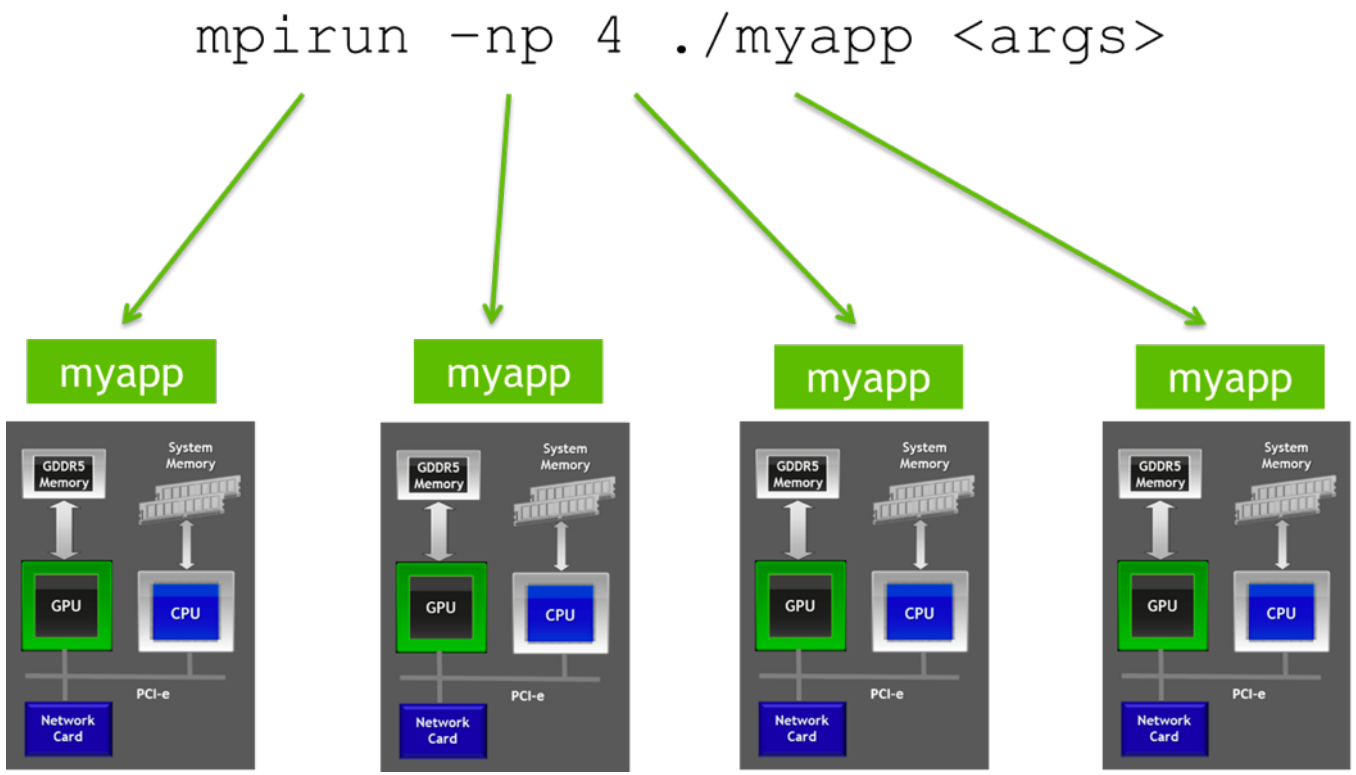
Specify partition gpu2

Specify number of gpus, like *gpu*:1

We will not run this example since the env_bootcamp does not have access to the gpu2 partition

How to submit MPI + GPU jobs?

CUDA Aware MPI



How to submit MPI + GPU jobs?

Compilation

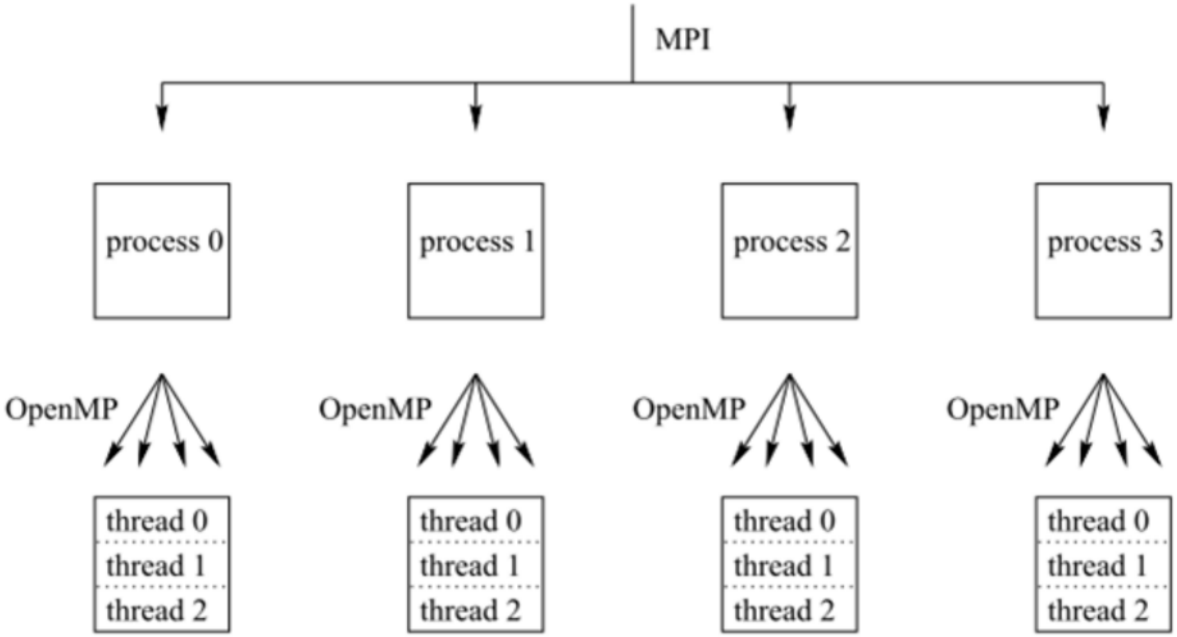
```
#!/bin/bash

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
#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
```

See Ex-5 for an example. Note we can not run this in this bootcamp

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
```

How to submit array based jobs

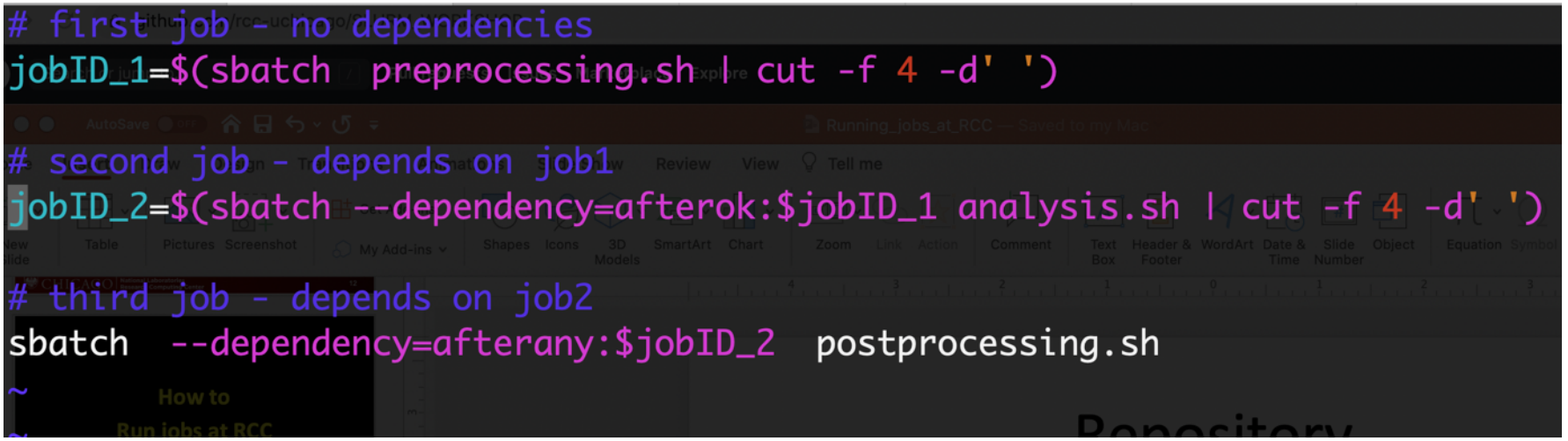
```
#!/bin/bash
# Job Name
#SBATCH --job-name=arrayjob
# Walltime requested
#SBATCH --time=0:10:00
#Add partition
#SBATCH --partition=edu

# 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

How to submit dependent jobs

SLURM Rule:

sbatch --dependency=type:job_id jobfile



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

How to submit Parallel batch jobs

```
#!/bin/sh
#SBATCH --time=01:00:00
#SBATCH --partition=edu
#SBATCH --ntasks=20
#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

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

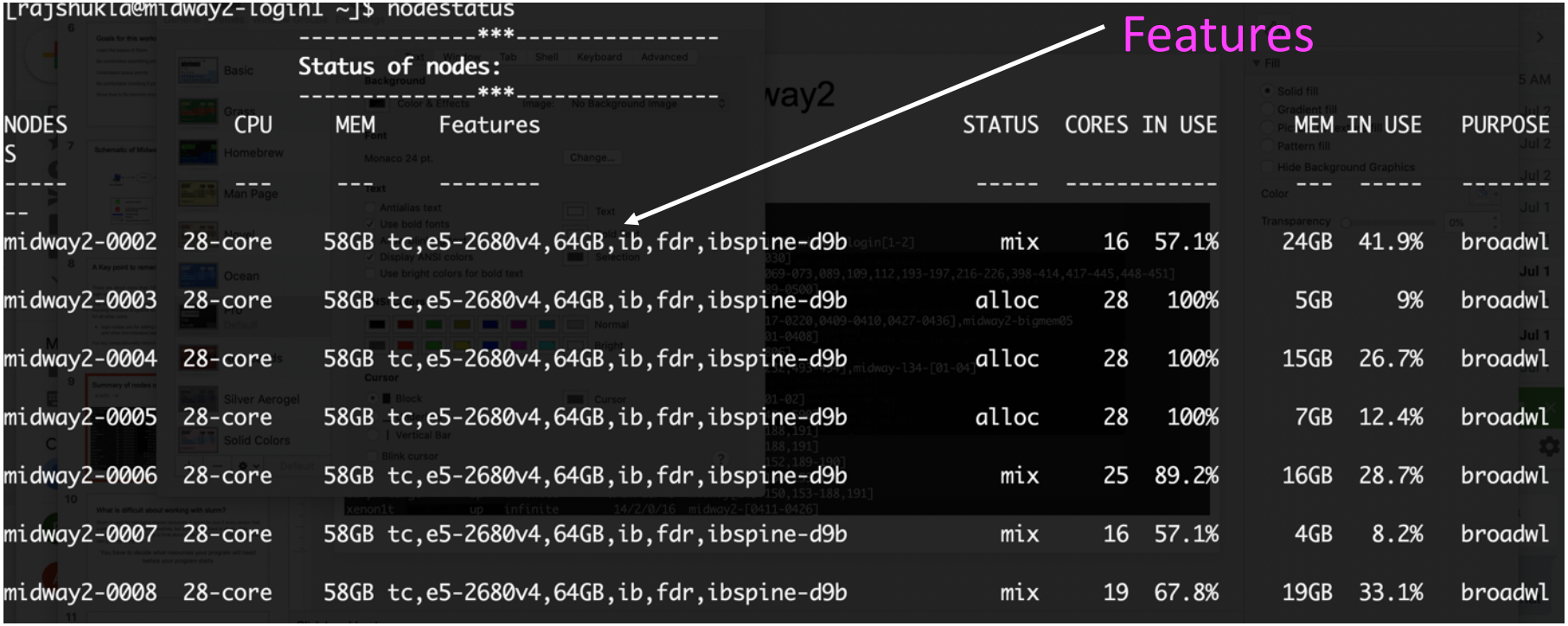
#SBATCH –exclusive

Add this in your batch script

#SBATCH –constraint=v100

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

nodestatus




```
[rajshukla@midway2-login1 ~]$ nodestatus
                    --------------***----------------
                    Status of nodes:
                    --------------***----------------
NODES                CPU     MEM      Features                                  STATUS  CORES IN USE     MEM IN USE    PURPOSE
S
-----               ---     ---      --------                                  -----  ------------     ------ -----    -------
--
midway2-0002   28-core    58GB tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b            mix      16  57.1%     24GB  41.9%    broadwl
midway2-0003   28-core    58GB tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b          alloc      28   100%      5GB     9%    broadwl
midway2-0004   28-core    58GB tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b          alloc      28   100%     15GB  26.7%    broadwl
midway2-0005   28-core    58GB tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b          alloc      28   100%      7GB  12.4%    broadwl
midway2-0006   28-core    58GB tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b            mix      25  89.2%     16GB  28.7%    broadwl
midway2-0007   28-core    58GB tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b            mix      16  57.1%      4GB   8.2%    broadwl
midway2-0008   28-core    58GB tc,e5-2680v4,64GB,ib,fdr,ibspine-d9b            mix      19  67.8%     19GB  33.1%    broadwl
```

What resources should I ask for?

This depends on the code you are running

GPUs If you code is build 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

You can fix these

- Node failure

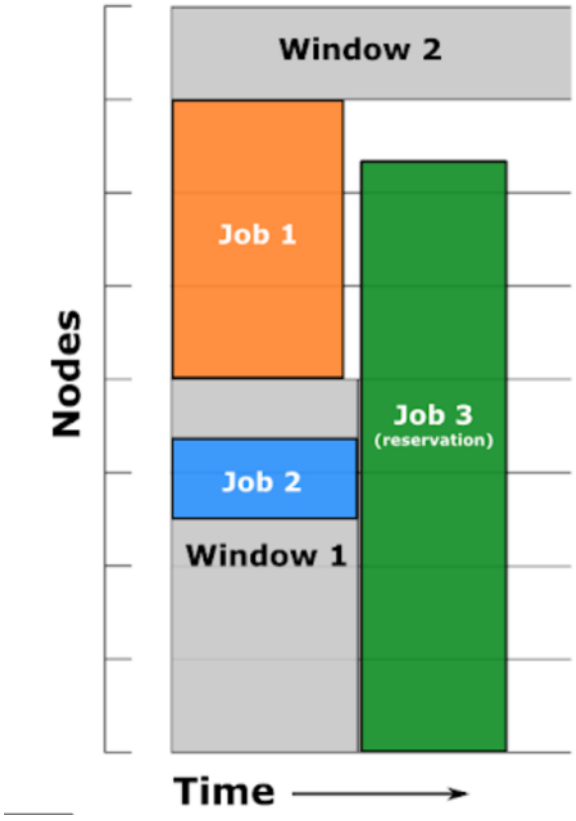
=> For node failure – please email help@rcc.uchicago.edu, if this happens

Job Priority

- Priority is calculates 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

the x axis is time

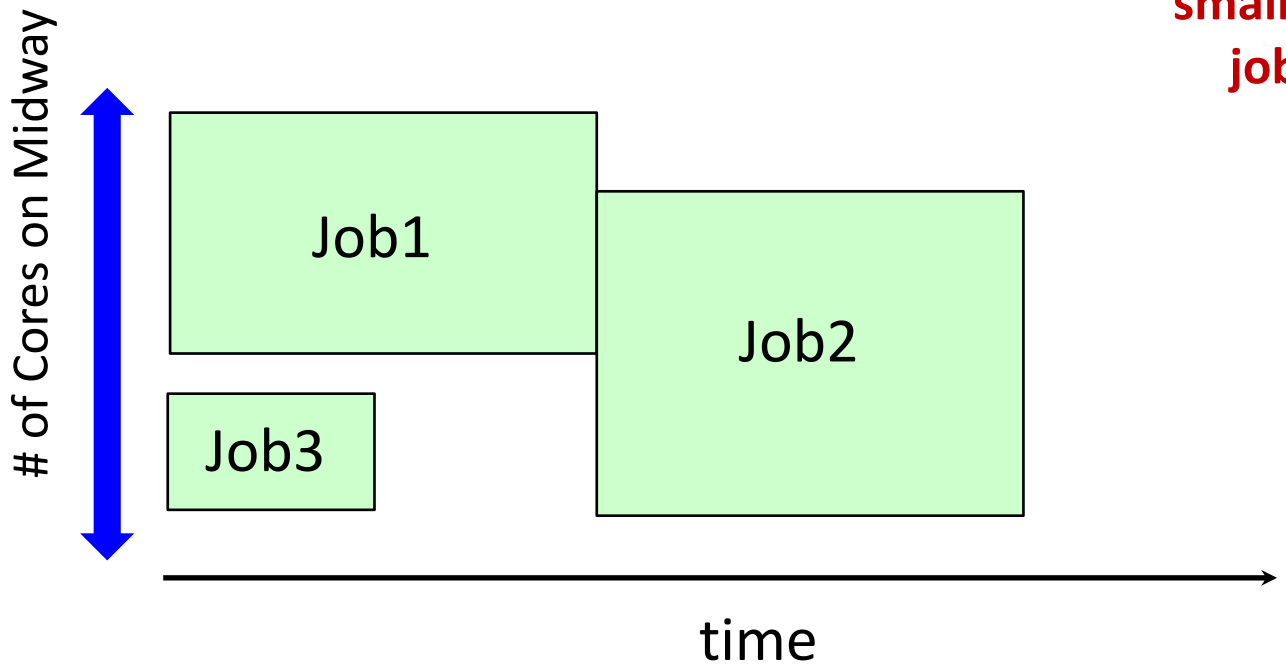
time

Job Priority

Backfill

Why has someone else's job started before mine?

The scheduler can fit smaller jobs in before big jobs are due to start

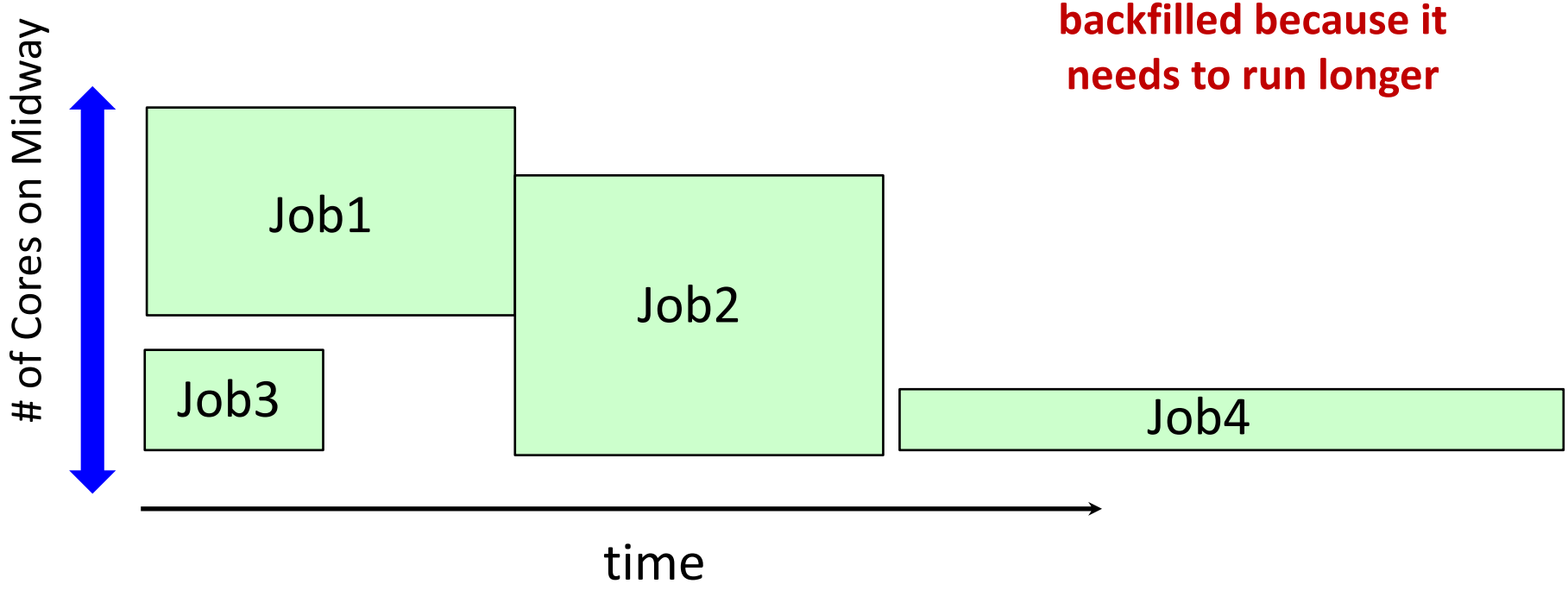


Job Priority

Backfill

Why has someone else's job started before mine?

Job4 while requesting less nodes, it can not be backfilled because it needs to run longer

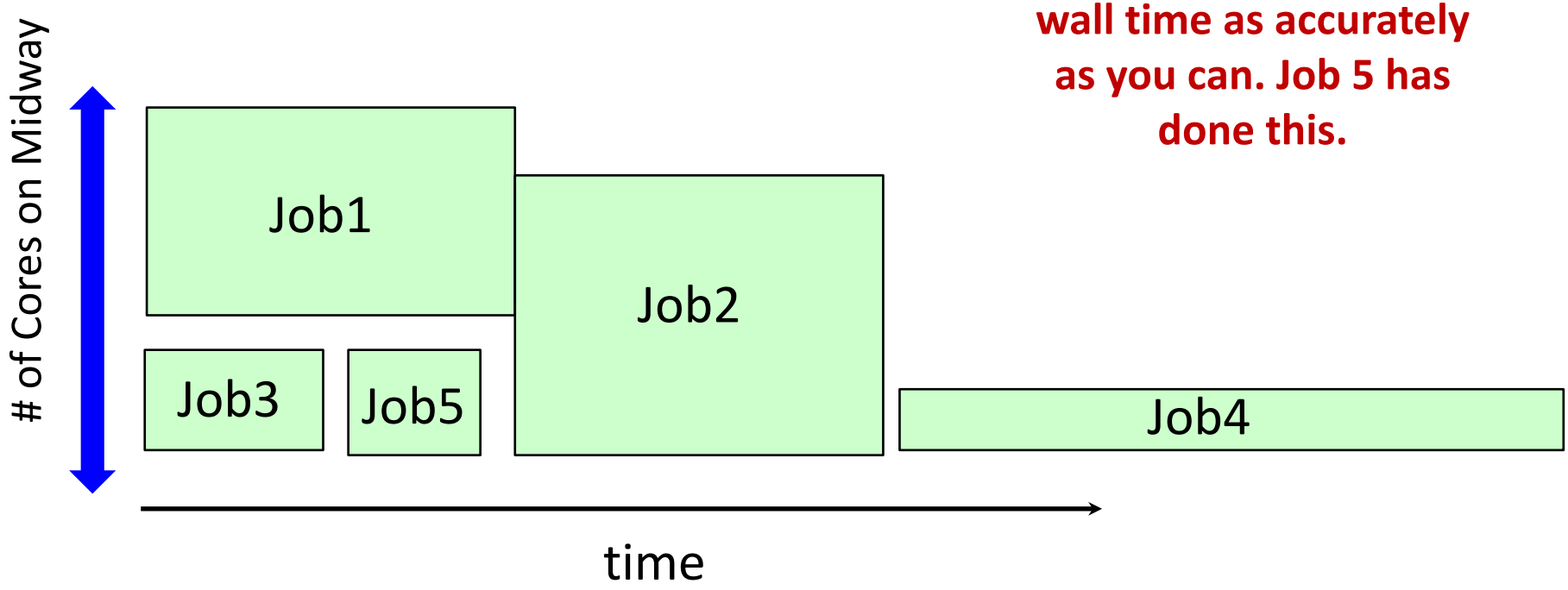


Job Priority

Backfill

Why has someone else's job started before mine?

You can take advantage of backfill by specifying wall time as accurately as you can. Job 5 has done this.

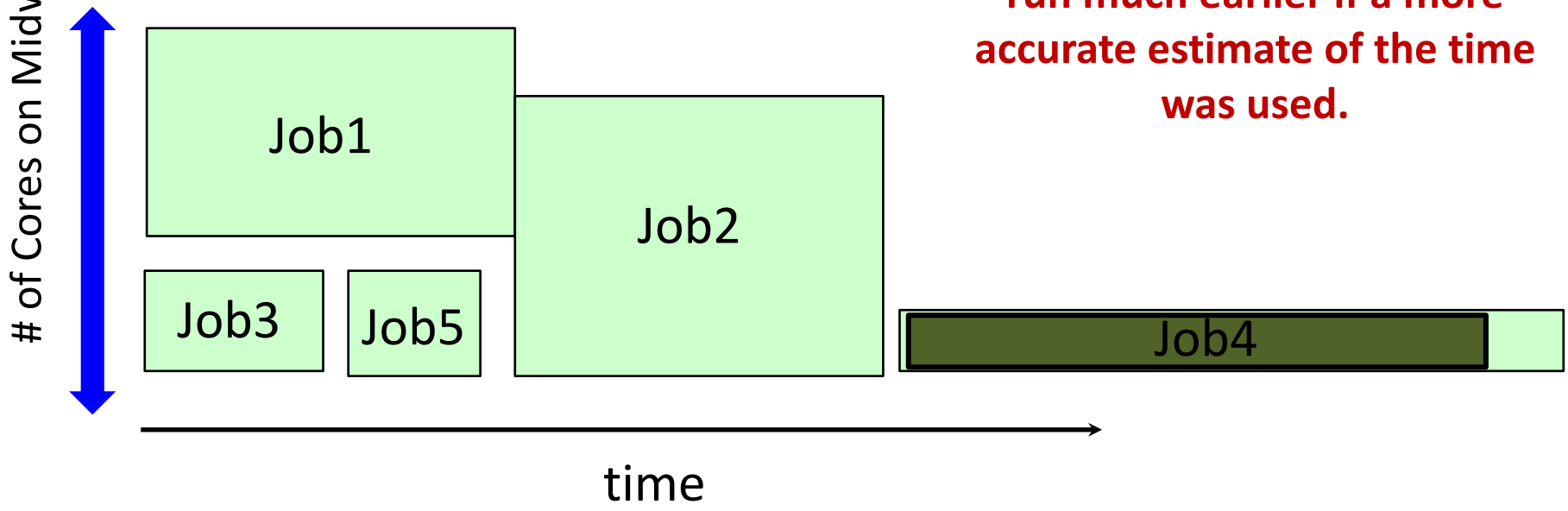


Job Priority

Backfill

Why has someone else's job started before mine?

You can take advantage of backfill by specifying wall time as accurately as you can. In this example job 4 could have run much earlier if a more accurate estimate of the time was used.



Job submission and monitoring

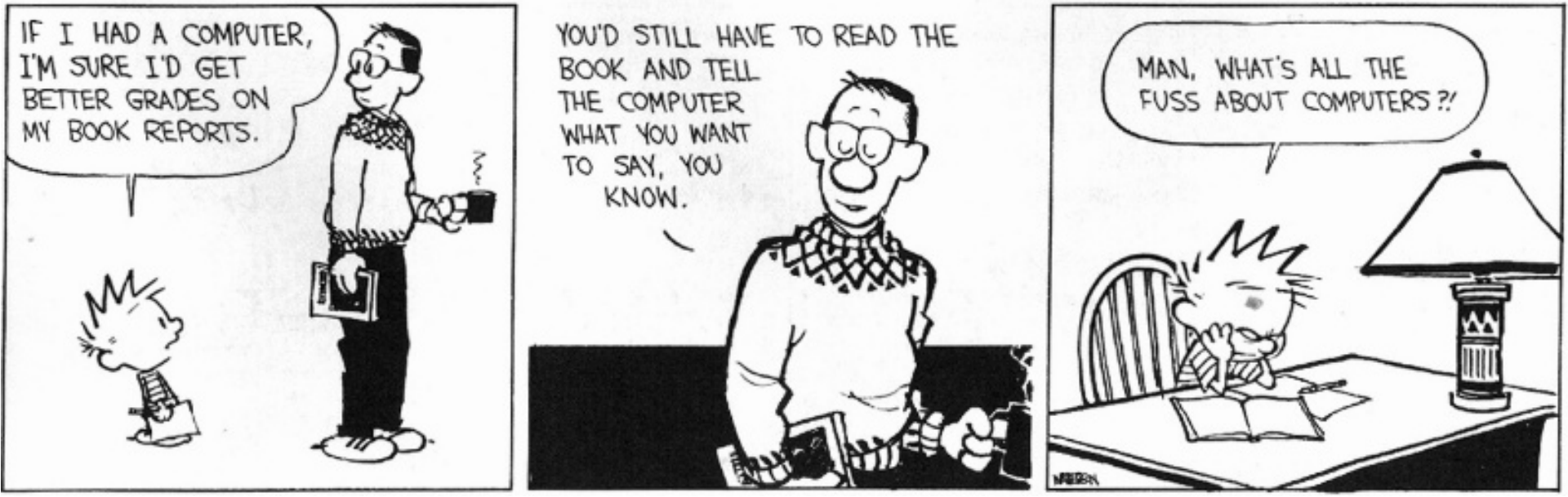
SLURM Commands

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

Recommended online resources

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

RCC Help



- Email: help@rcc.uchicago.edu
- Web: rcc.uchicago.edu
- Phone: 773-795-2667
- Walk-in: Regenstein Library, suite 216

END OF SLIDE SHOW