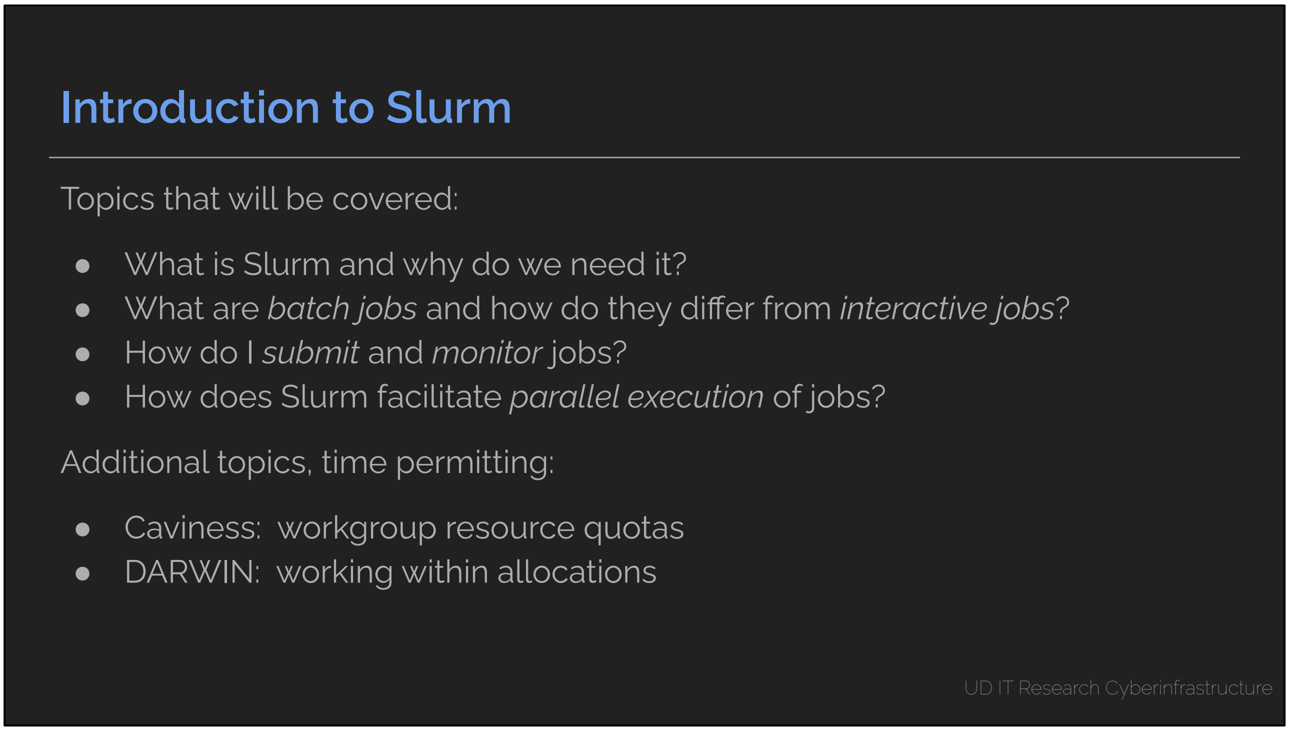
UD IT Research Cyberinfrastructure

Introduction to Slurm



Introduction to Slurm

Topics that will be covered:

- What is Slurm and why do we need it?
- What are *batch jobs* and how do they differ from *interactive jobs*?
- How do I *submit* and *monitor* jobs?
- How does Slurm facilitate *parallel execution* of jobs?

Additional topics, time permitting:

- Caviness: workgroup resource quotas
- DARWIN: working within allocations

What is Slurm and why do we need it?

Given the limited capacity of the wheelbarrow, how does one move the rocks?

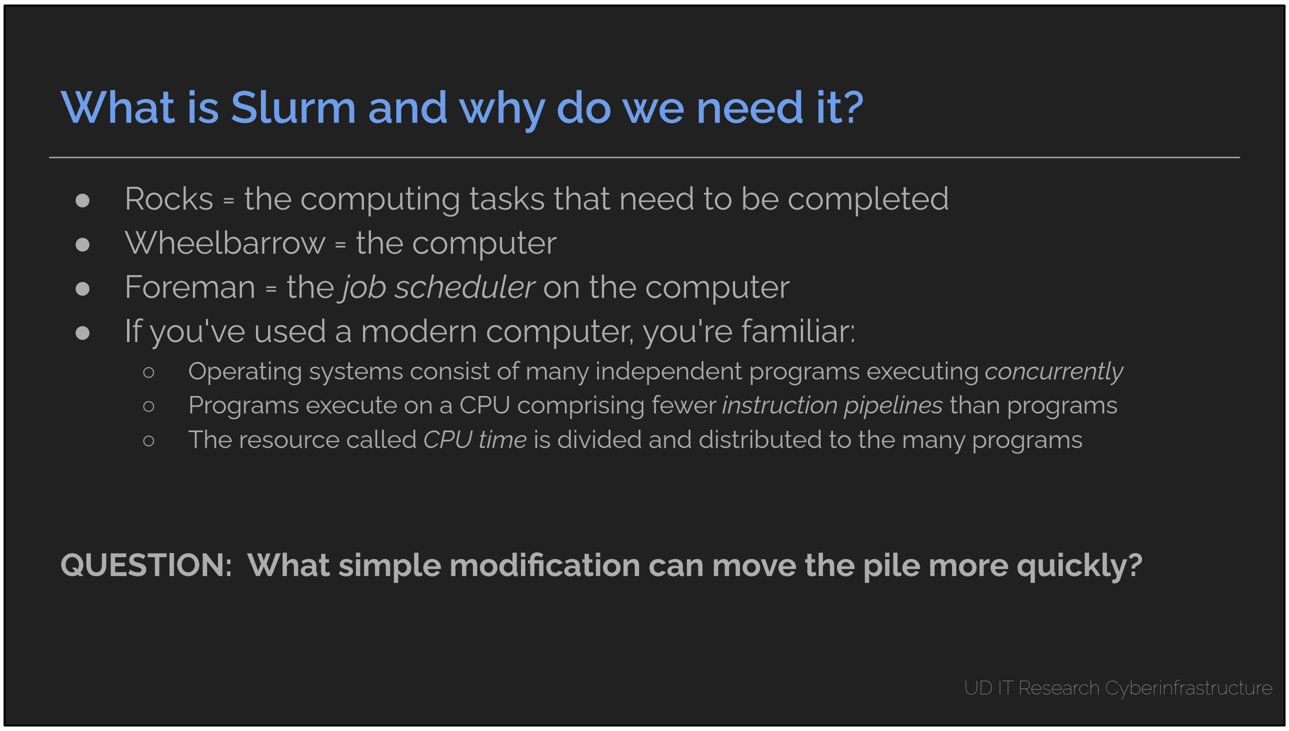
UD IT Research Cyberinfrastructure

A standard menial task: moving rocks.

What is Slurm and why do we need it?

Given the limited capacity of the wheelbarrow, how does one move the rocks?

- Transfer rocks from outside of pile into wheelbarrow
  - Many unique shapes/sizes, so count per load will vary
- When full, roll wheelbarrow to destination, dump
- Repeat until all rocks have been moved



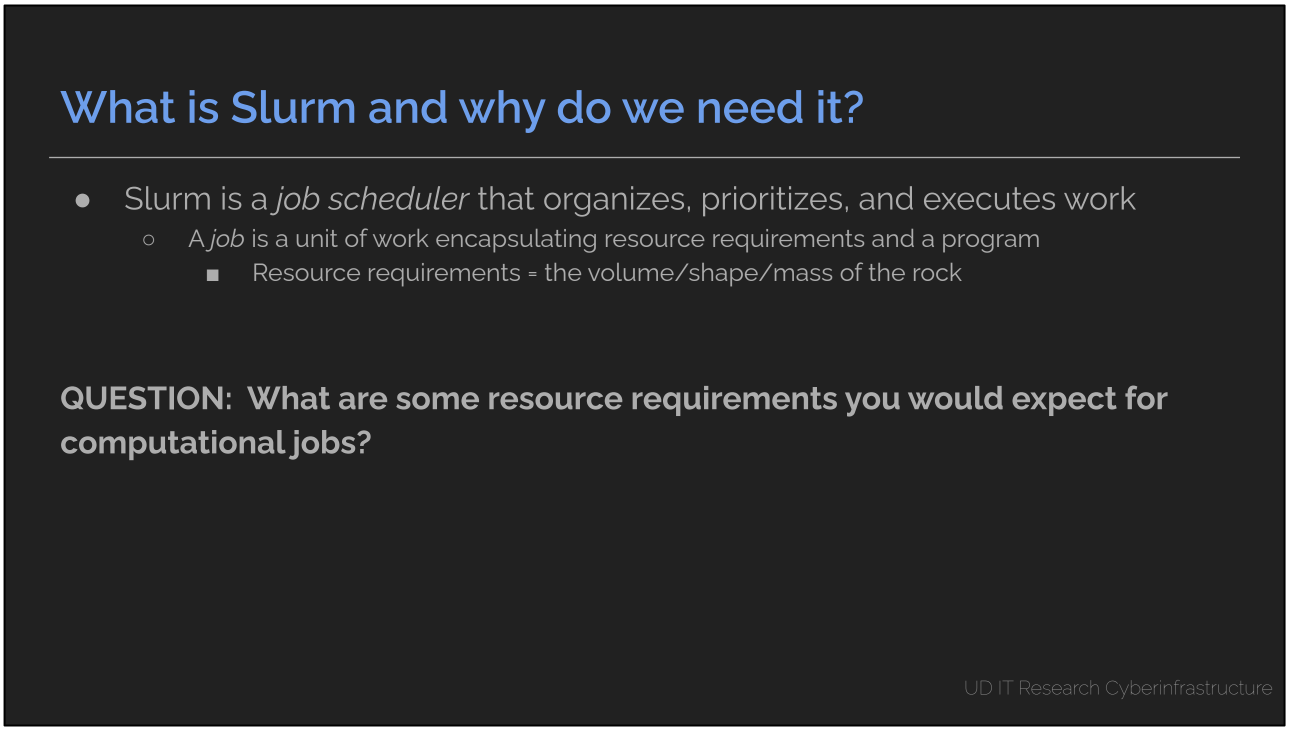
What is Slurm and why do we need it?

- Rocks = the computing tasks that need to be completed
- Wheelbarrow = the computer
- Foreman = the *job scheduler* on the computer
- If you've used a modern computer, you're familiar:
  - Operating systems consist of many independent programs executing *concurrently*
  - Programs execute on a CPU comprising fewer *instruction pipelines* than programs
  - The resource called *CPU time* is divided and distributed to the many programs

QUESTION: What simple modification can move the pile more quickly?

UD IT Research Cyberinfrastructure

What sub-component of a modern CPU is associated with an instruction pipeline?
CPU time is associated with each instruction pipeline, so a 32-core CPU runs
32-times faster than real time elapses — real time is called wall time.
ANSWER: add more wheelbarrows

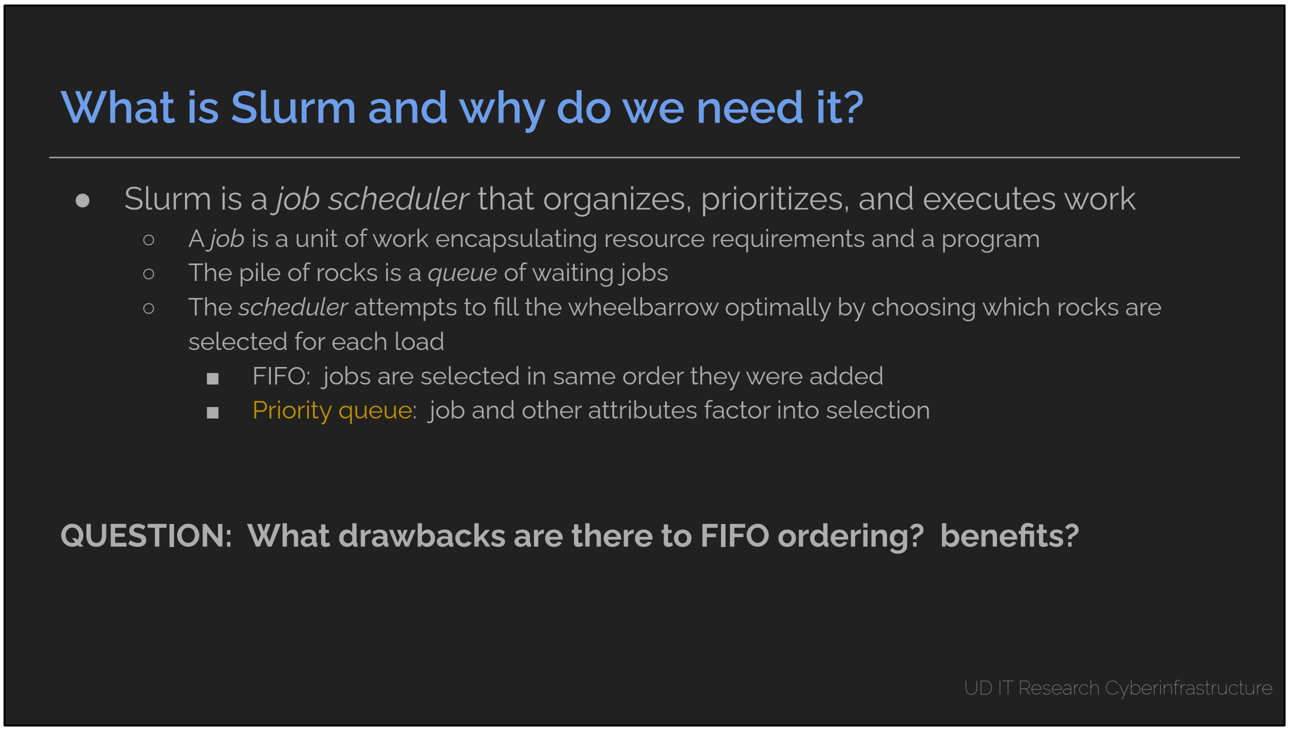


What is Slurm and why do we need it?

- Slurm is a *job scheduler* that organizes, prioritizes, and executes work
  - A *job* is a unit of work encapsulating resource requirements and a program
    - Resource requirements = the volume/shape/mass of the rock

QUESTION: What are some resource requirements you would expect for computational jobs?

ANSWER: CPU/cores, memory, time, network bandwidth, scratch storage, …

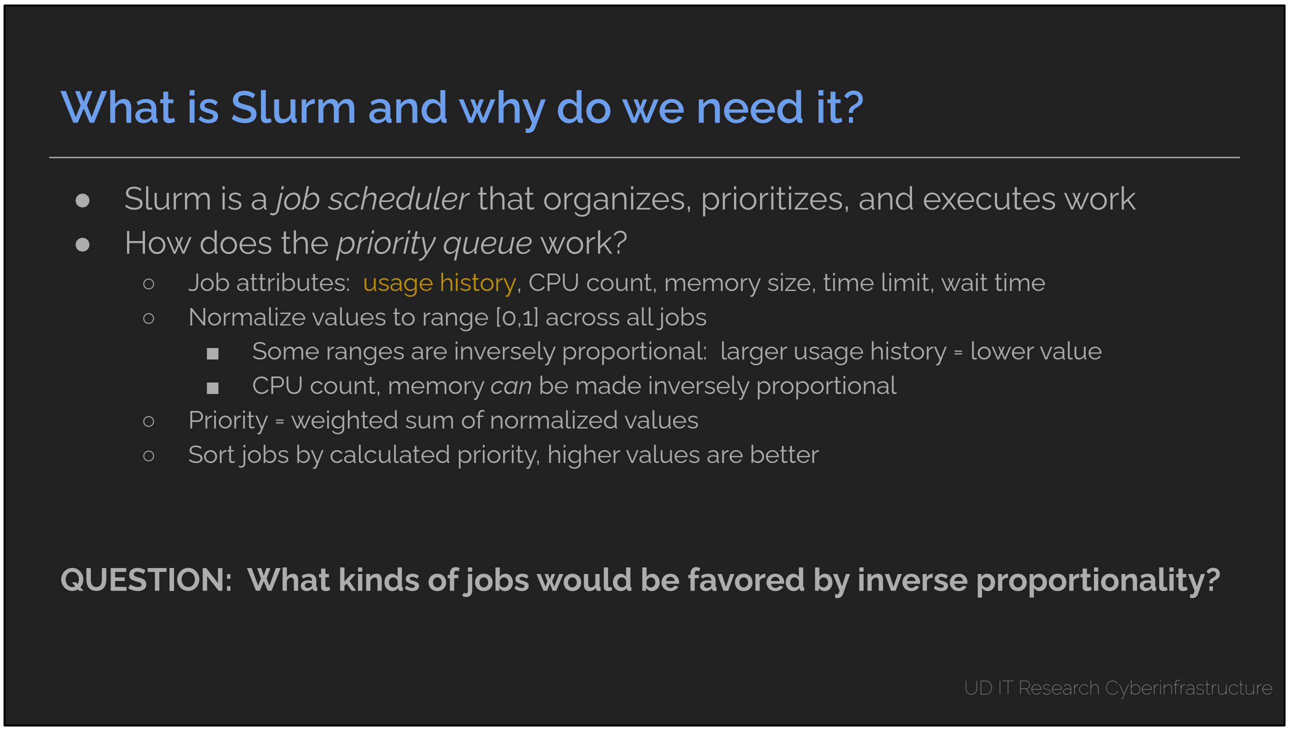


What is Slurm and why do we need it?

- Slurm is a *job scheduler* that organizes, prioritizes, and executes work
  - A *job* is a unit of work encapsulating resource requirements and a program
  - The pile of rocks is a *queue* of waiting jobs
  - The *scheduler* attempts to fill the wheelbarrow optimally by choosing which rocks are selected for each load
    - FIFO: jobs are selected in same order they were added
    - Priority queue: job and other attributes factor into selection

QUESTION: What drawbacks are there to FIFO ordering? benefits?

Slurm has a single queue containing all jobs; other schedulers (like Grid Engine or PBS) offer a hierarchy of one or more queues.
ANSWER: unused resources and delay — large gaps between big rocks that could be filled with smaller stones; very clear order to selection of a job



What is Slurm and why do we need it?

- Slurm is a *job scheduler* that organizes, prioritizes, and executes work
- How does the *priority queue* work?
  - Job attributes: usage history, CPU count, memory size, time limit, wait time
  - Normalize values to range [0,1] across all jobs
    - Some ranges are inversely proportional: larger usage history = lower value
    - CPU count, memory *can* be made inversely proportional
  - Priority = weighted sum of normalized values
  - Sort jobs by calculated priority, higher values are better

QUESTION: What kinds of jobs would be favored by inverse proportionality?

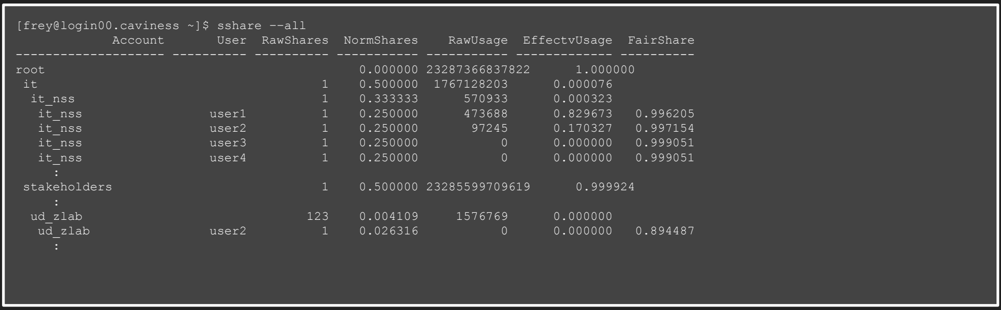
UD IT Research Cyberinfrastructure

ANSWER: Jobs requesting LOWER CPU counts, memory sizes — "small" jobs

What is Slurm and why do we need it?

USAGE HISTORY

As jobs are executed, a user's resource usage is aggregated. Time since a job completed weights its contribution (older jobs = smaller contribution).



```
[frey@login00.caviness ~]$ sshare --all
             Account       User  RawShares  NormShares    RawUsage  EffectvUsage  FairShare
-------------------- ---------- ---------- ----------- ----------- ------------- ----------
root                                          0.000000 23287366837822      1.000000
 it                                      1    0.500000  1767128203      0.000076
  it_nss                                 1    0.333333      570933      0.000323
   it_nss                 user1          1    0.250000      473688      0.829673   0.996205
   it_nss                 user2          1    0.250000       97245      0.170327   0.997154
   it_nss                 user3          1    0.250000           0      0.000000   0.999051
   it_nss                 user4          1    0.250000           0      0.000000   0.999051
     :
 stakeholders                            1    0.500000 23285599709619      0.999924
     :
  ud_zlab                              123    0.004109     1576769      0.000000
   ud_zlab                user2          1    0.026316           0      0.000000   0.894487
     :
```

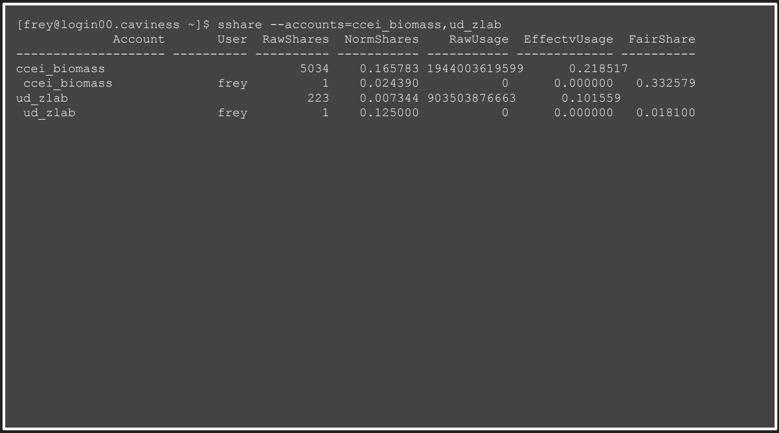
UD IT Research Cyberinfrastructure

Two top-level tiers with equal shares: "it" and "stakeholders"; four users in "it_nss" with equal shares
Each stakeholder workgroup on Caviness is given shares equating with investment in the cluster
Raw usage is the last-calculated aggregate usage; shares weight the fraction of "total priority" associated with each tier

What is Slurm and why do we need it?

USAGE HISTORY

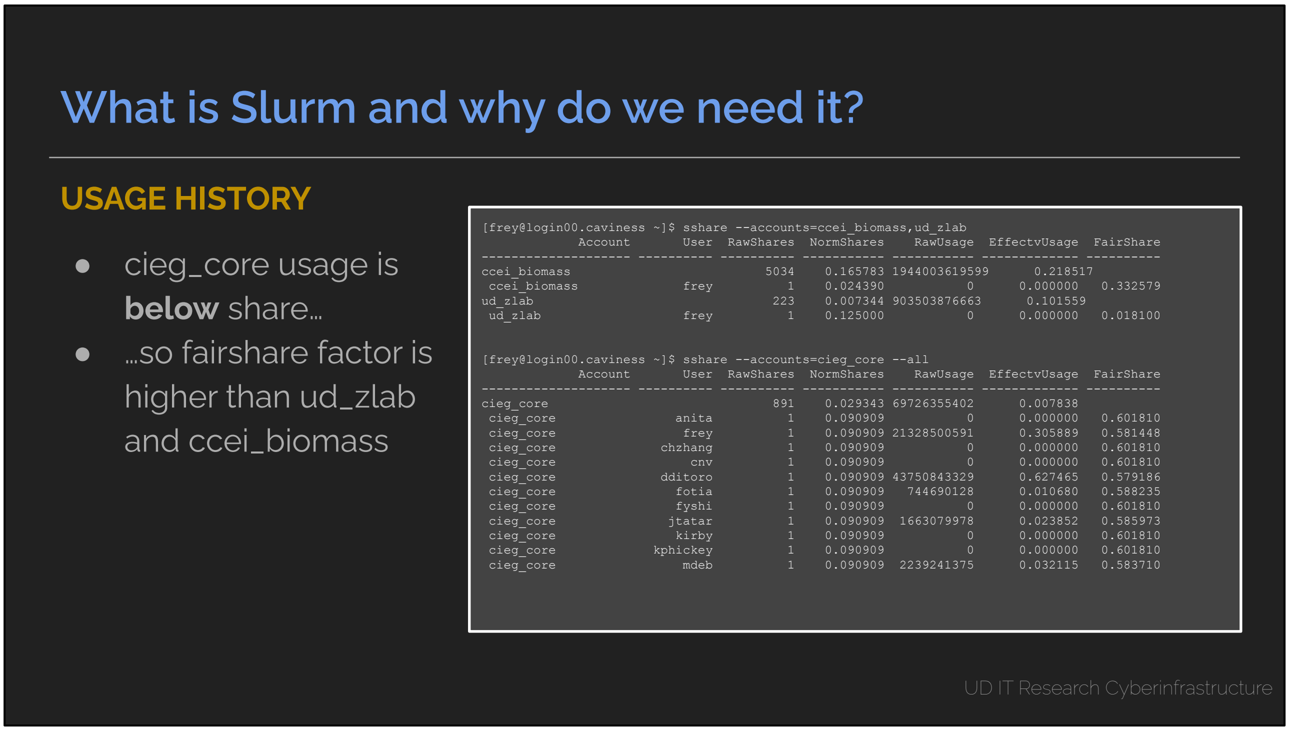
- Effective usage **exceeds** workgroup normalized shares…
- …but ud_zlab (13.8x) ≫ ccei_biomass (1.3x)



```
[frey@login00.caviness ~]$ sshare --accounts=ccei_biomass,ud_zlab
             Account       User  RawShares  NormShares    RawUsage  EffectvUsage  FairShare
-------------------- ---------- ---------- ----------- ----------- ------------- ----------
ccei_biomass                          5034    0.165783 1944003619599      0.218517
 ccei_biomass              frey          1    0.024390           0      0.000000   0.332579
ud_zlab                                223    0.007344 903503876663      0.101559
 ud_zlab                   frey          1    0.125000           0      0.000000   0.018100
```

UD IT Research Cyberinfrastructure

Relative to ud_zlab, jobs for workgroup ccei_biomass should execute sooner — thus fairshare is lower



What is Slurm and why do we need it?

USAGE HISTORY

- cieg_core usage is **below** share…
- …so fairshare factor is higher than ud_zlab and ccei_biomass

```
[frey@login00.caviness ~]$ sshare --accounts=ccei_biomass,ud_zlab
             Account       User  RawShares  NormShares    RawUsage  EffectvUsage  FairShare
-------------------- ---------- ---------- ----------- ----------- ------------- ----------
ccei_biomass                          5034    0.165783 1944003619599      0.218517
 ccei_biomass              frey          1    0.024390           0      0.000000   0.332579
ud_zlab                                223    0.007344 903503876663      0.101559
 ud_zlab                   frey          1    0.125000           0      0.000000   0.018100


[frey@login00.caviness ~]$ sshare --accounts=cieg_core --all
             Account       User  RawShares  NormShares    RawUsage  EffectvUsage  FairShare
-------------------- ---------- ---------- ----------- ----------- ------------- ----------
cieg_core                              891    0.029343 69726355402      0.007838
 cieg_core                anita          1    0.090909           0      0.000000   0.601810
 cieg_core                 frey          1    0.090909 21328500591      0.305889   0.581448
 cieg_core              chzhang          1    0.090909           0      0.000000   0.601810
 cieg_core                  cnv          1    0.090909           0      0.000000   0.601810
 cieg_core              dditoro          1    0.090909 43750843329      0.627465   0.579186
 cieg_core                fotia          1    0.090909   744690128      0.010680   0.588235
 cieg_core                fyshi          1    0.090909           0      0.000000   0.601810
 cieg_core               jtatar          1    0.090909  1663079978      0.023852   0.585973
 cieg_core                kirby          1    0.090909           0      0.000000   0.601810
 cieg_core             kphickey          1    0.090909           0      0.000000   0.601810
 cieg_core                 mdeb          1    0.090909  2239241375      0.032115   0.583710
```

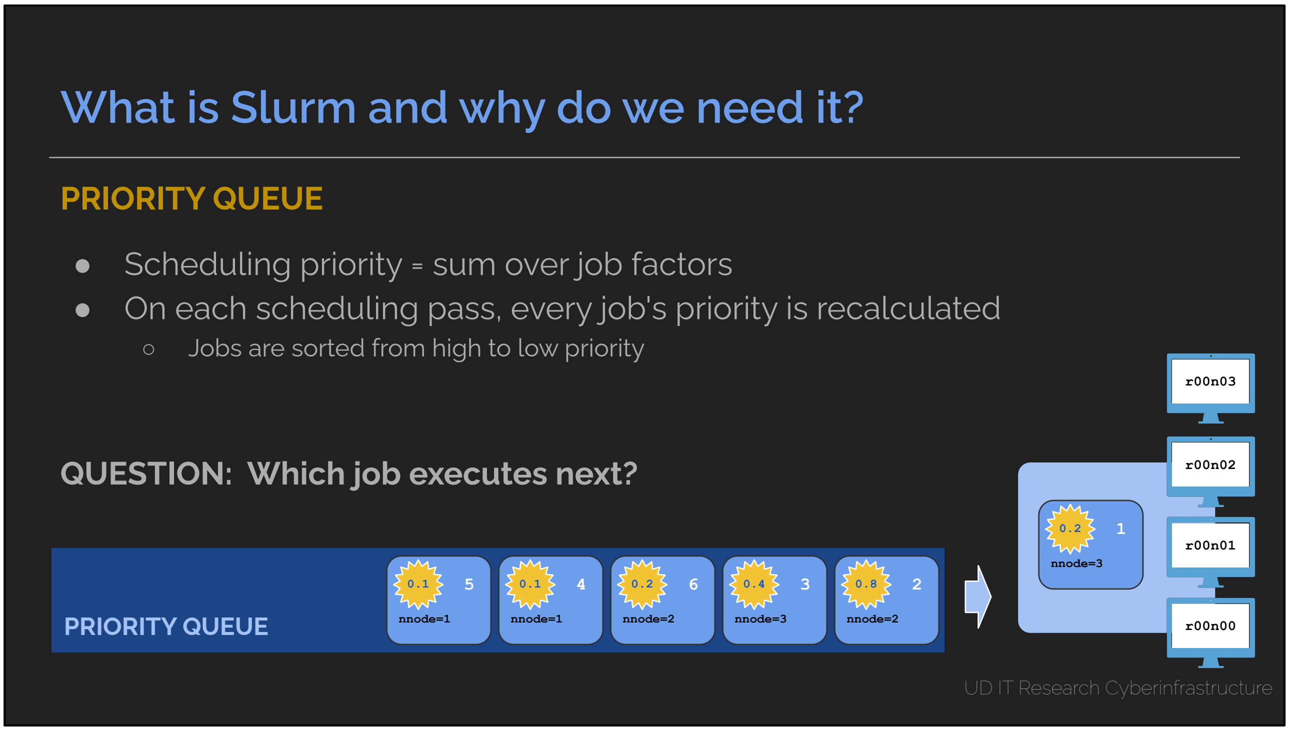
UD IT Research Cyberinfrastructure

Note that an individual user will have unique fairshare factors based on the workgroup used to submit the job
Also note that an individual's usage under a workgroup increases usage by the entire workgroup and thus affects fairshare factor for all members

What is Slurm and why do we need it?

PRIORITY QUEUE

- Scheduling priority = sum over job factors
  - Caviness: `0xC0000000(QoS) + 0x3FFFF000(FairShare) + 0x999(Age) + 0x333(#CPU) + 0x147(#Node) + 0xF5(Mem) + 0xF5(#GPU)`
    - Factors being weighted are in range [0,1], so:
      - Min priority = `0x00000000 + 0x00000000 + 0x000 + 0x000 + 0x000 + 0x00 + 0x00 = 0x00000000`
      - Max priority = `0xC0000000 + 0x3FFFF000 + 0x999 + 0x333 + 0x147 + 0xF5 + 0xF5 = 0xFFFFFFFD`
  - DARWIN: `0xC0000000(QoS) + 0x3FFFF000(FairShare) + 0x999(Age) + 0x233(#CPU) + 0x233(Mem) + 0x100(#GPU) + 0x100(#Node)`
    - Factors being weighted are in range [0,1], so:
      - Min priority = `0x00000000 + 0x00000000 + 0x000 + 0x000 + 0x000 + 0x00 + 0x00 = 0x00000000`
      - Max priority = `0xC0000000 + 0x3FFFF000 + 0x999 + 0x233 + 0x233 + 0x100 + 0x100 = 0xFFFFFFFF`

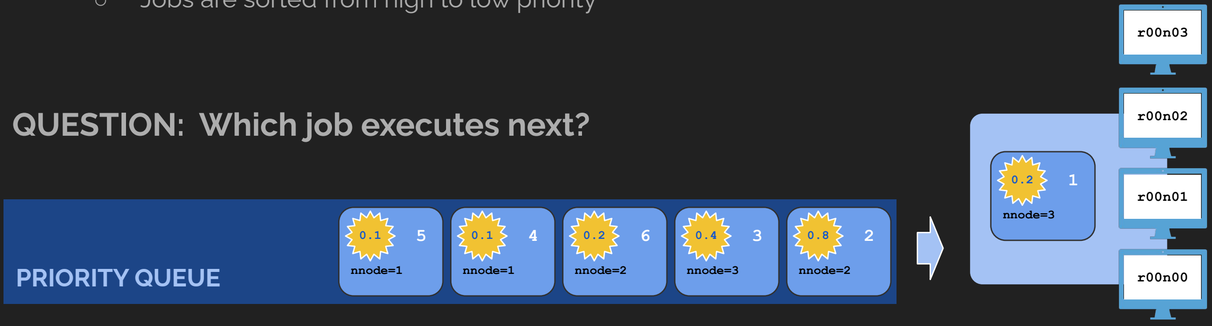


What is Slurm and why do we need it?

PRIORITY QUEUE

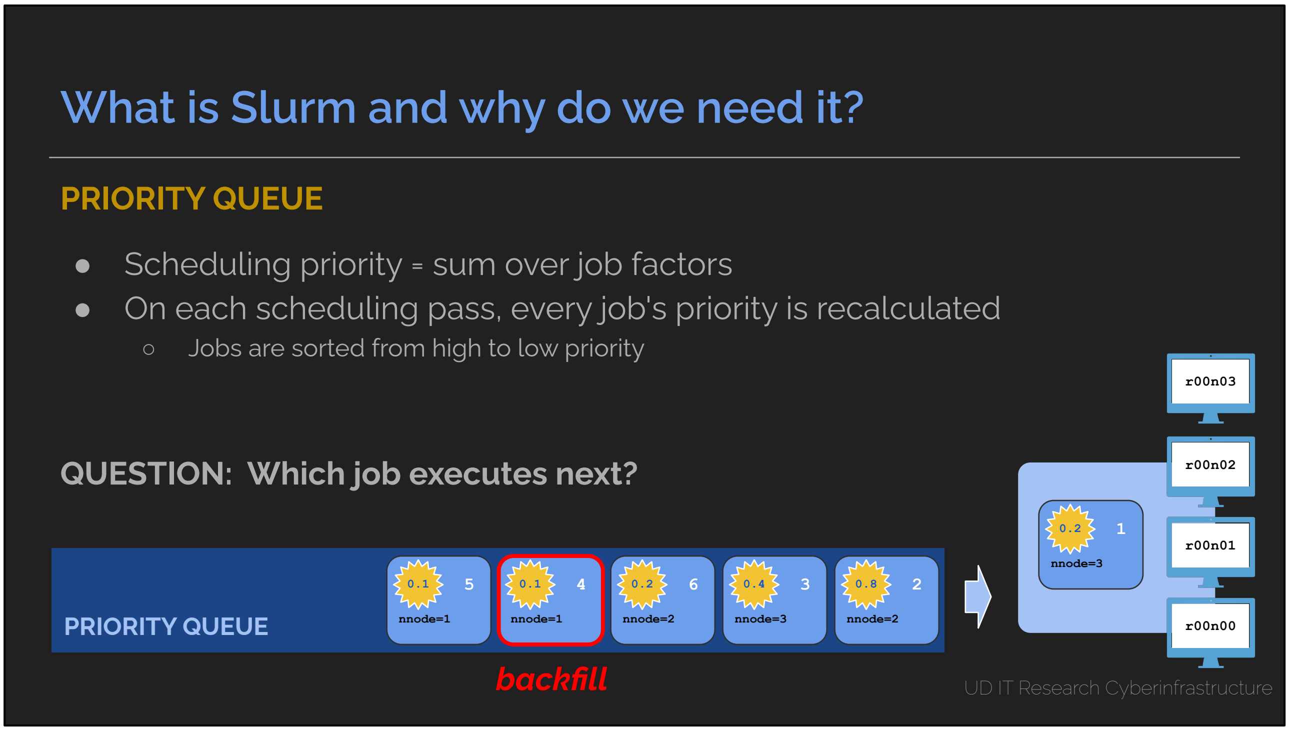
- Scheduling priority = sum over job factors
- On each scheduling pass, every job's priority is recalculated
  - Jobs are sorted from high to low priority

QUESTION: Which job executes next?



UD IT Research Cyberinfrastructure

ANSWER: FIFO would wait for job 1 to finish before starting job 2; but that leaves nodes unused for the duration of jobs 1, 2, 3, and 6



What is Slurm and why do we need it?

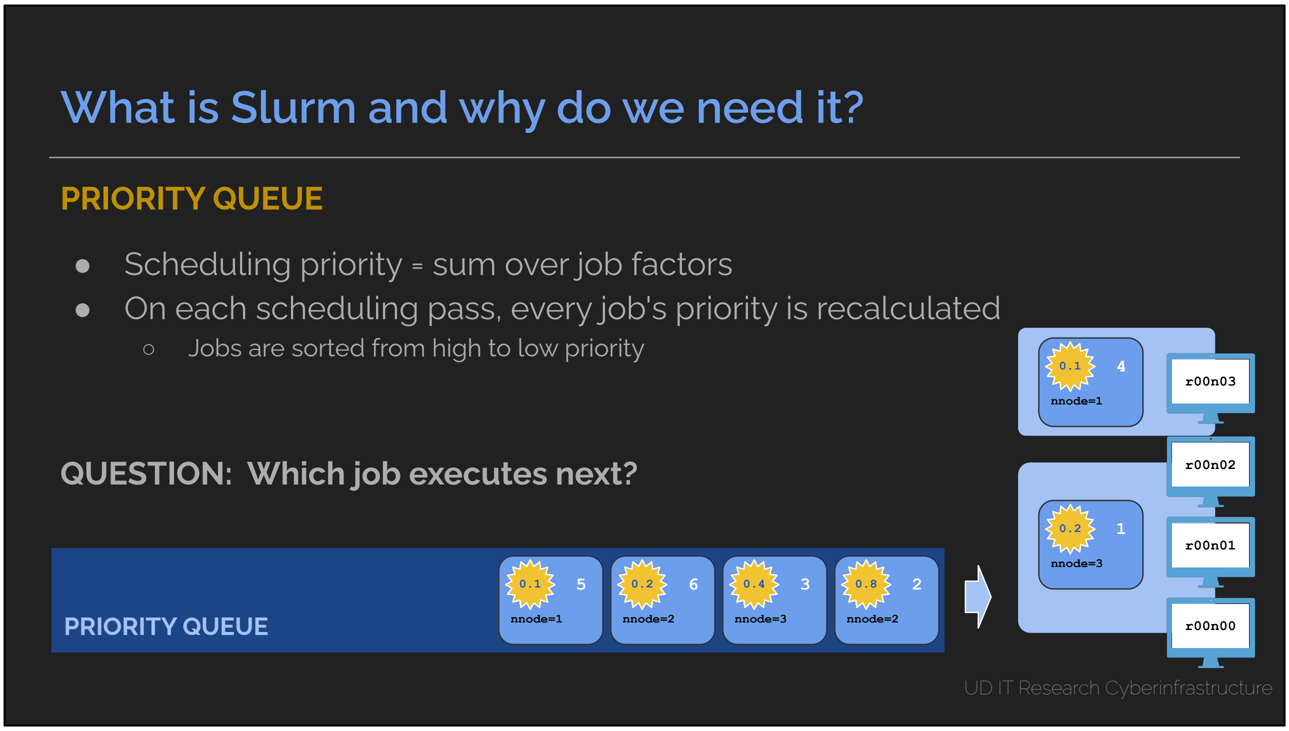
PRIORITY QUEUE

- Scheduling priority = sum over job factors
- On each scheduling pass, every job's priority is recalculated
  - Jobs are sorted from high to low priority

QUESTION: Which job executes next?

UD IT Research Cyberinfrastructure

ANSWER: Backfill can override priority to increase active use of resources

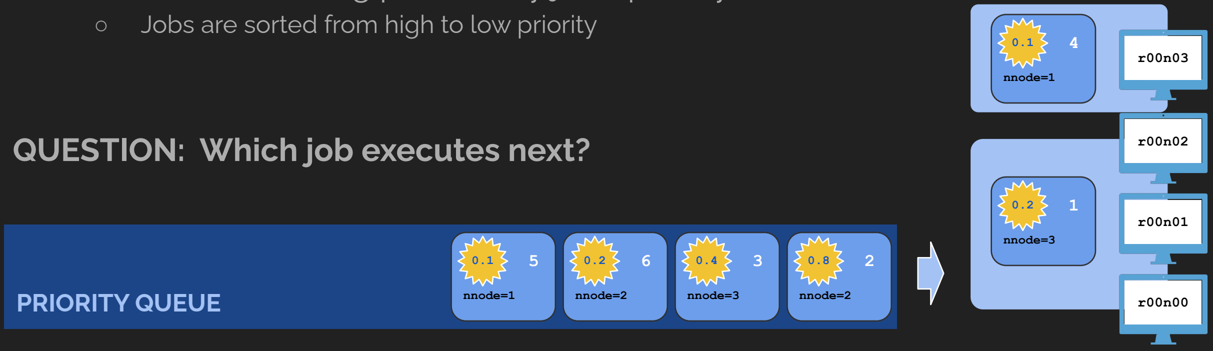


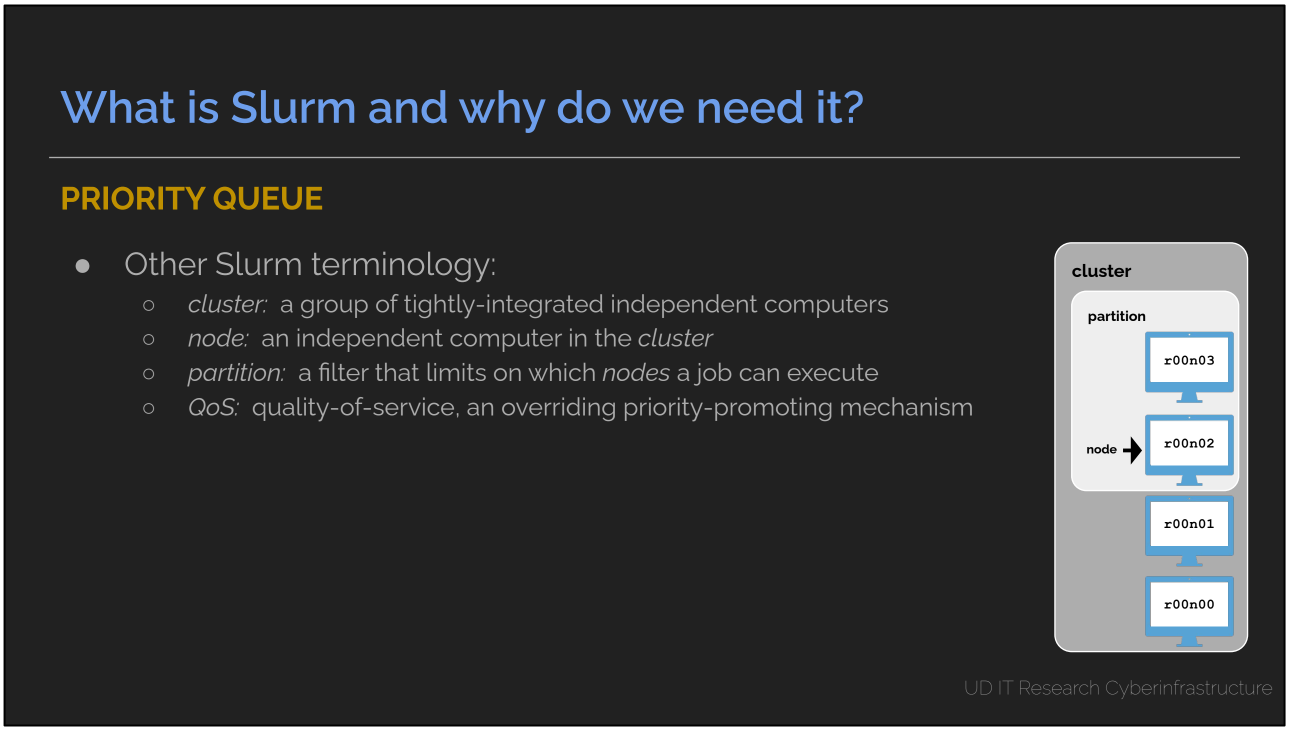
What is Slurm and why do we need it?

PRIORITY QUEUE

- Scheduling priority = sum over job factors
- On each scheduling pass, every job's priority is recalculated
  - Jobs are sorted from high to low priority

QUESTION: Which job executes next?

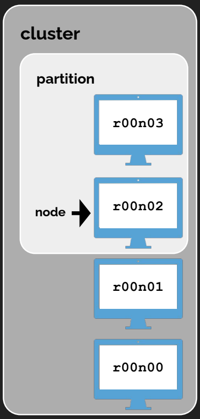




What is Slurm and why do we need it?

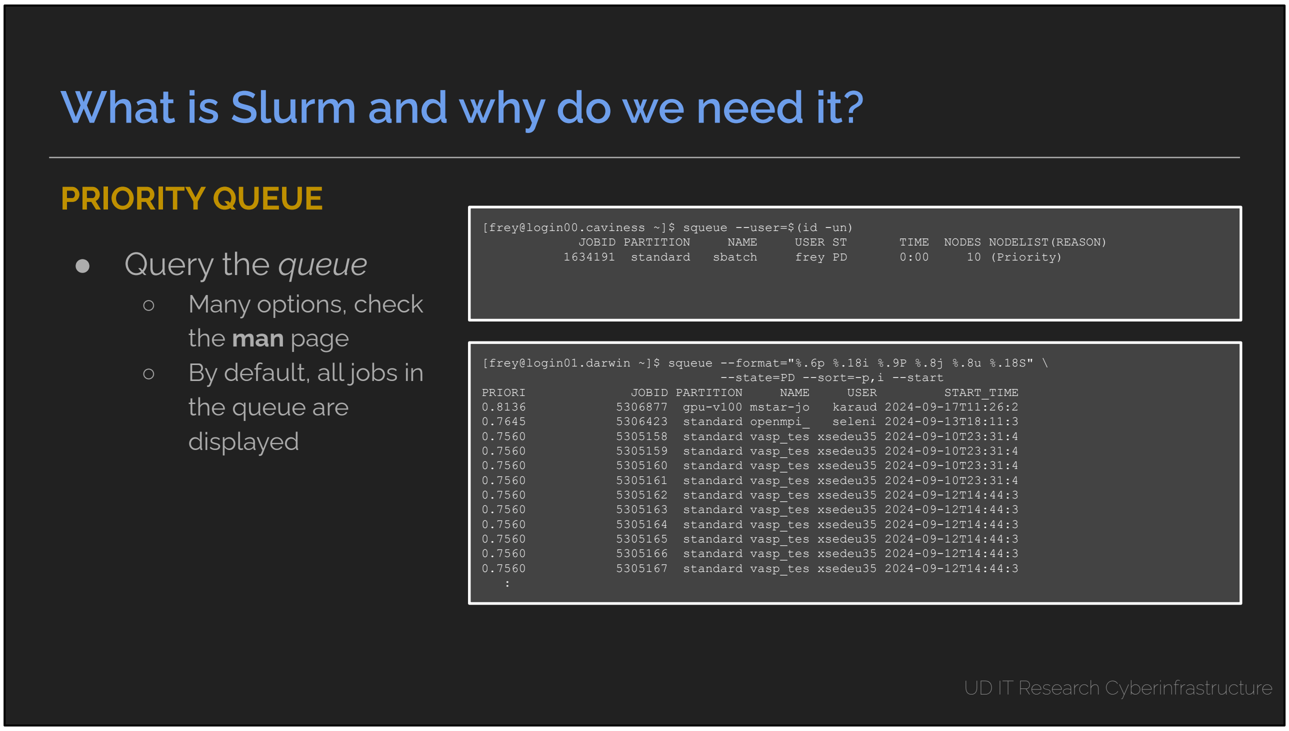
PRIORITY QUEUE

- Other Slurm terminology:
  - *cluster:* a group of tightly-integrated independent computers
  - *node:* an independent computer in the *cluster*
  - *partition:* a filter that limits on which *nodes* a job can execute
  - *QoS:* quality-of-service, an overriding priority-promoting mechanism



UD IT Research Cyberinfrastructure

Recall that the weight on QoS in the priority formula was the highest weighting value



What is Slurm and why do we need it?

PRIORITY QUEUE

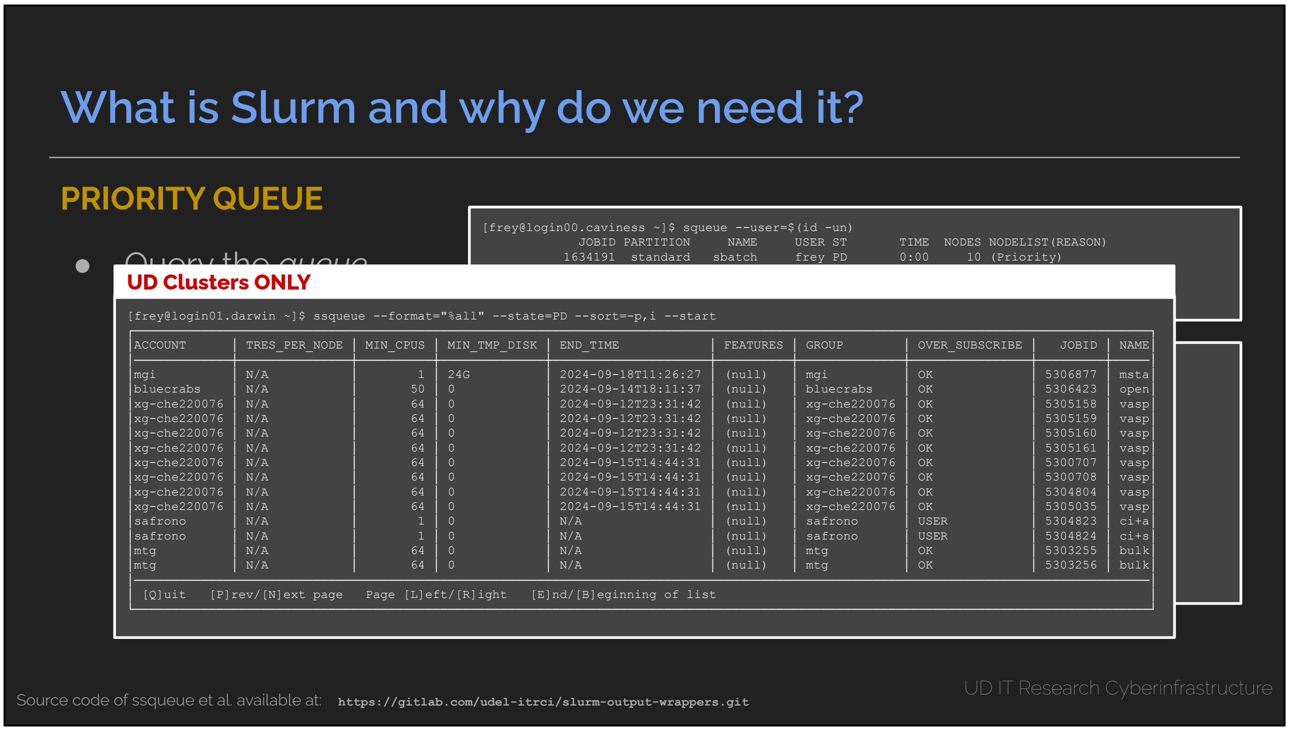
- Query the *queue*
  - Many options, check the **man** page
  - By default, all jobs in the queue are displayed

```
[frey@login00.caviness ~]$ squeue --user=$(id -un)
             JOBID  PARTITION      NAME      USER ST        TIME  NODES NODELIST(REASON)
           1634191   standard    sbatch      frey PD        0:00     10 (Priority)
```

```
[frey@login01.darwin ~]$ squeue --format="%.6p %.18i %.9P %.8j %.8u %.18S" \
                                --state=PD --sort=-p,i --start
PRIORI              JOBID PARTITION     NAME     USER         START_TIME
0.8136            5306877  gpu-v100 mstar-jo   karaud 2024-09-17T11:26:2
0.7645            5306423  standard openmpi_   seleni 2024-09-13T18:11:3
0.7560            5305158  standard vasp_tes xsedeu35 2024-09-10T23:31:4
0.7560            5305159  standard vasp_tes xsedeu35 2024-09-10T23:31:4
0.7560            5305160  standard vasp_tes xsedeu35 2024-09-10T23:31:4
0.7560            5305161  standard vasp_tes xsedeu35 2024-09-10T23:31:4
0.7560            5305162  standard vasp_tes xsedeu35 2024-09-12T14:44:3
0.7560            5305163  standard vasp_tes xsedeu35 2024-09-12T14:44:3
0.7560            5305164  standard vasp_tes xsedeu35 2024-09-12T14:44:3
0.7560            5305165  standard vasp_tes xsedeu35 2024-09-12T14:44:3
0.7560            5305166  standard vasp_tes xsedeu35 2024-09-12T14:44:3
0.7560            5305167  standard vasp_tes xsedeu35 2024-09-12T14:44:3
   :
```

UD IT Research Cyberinfrastructure

Difference between the first and second = jobs in all states for current user versus pending jobs for all users



What is Slurm and why do we need it?

PRIORITY QUEUE

- Query the *queue*

```
[frey@login00.caviness ~]$ squeue --user=$(id -un)
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1634191  standard   sbatch     frey PD       0:00     10 (Priority)
```

UD Clusters ONLY

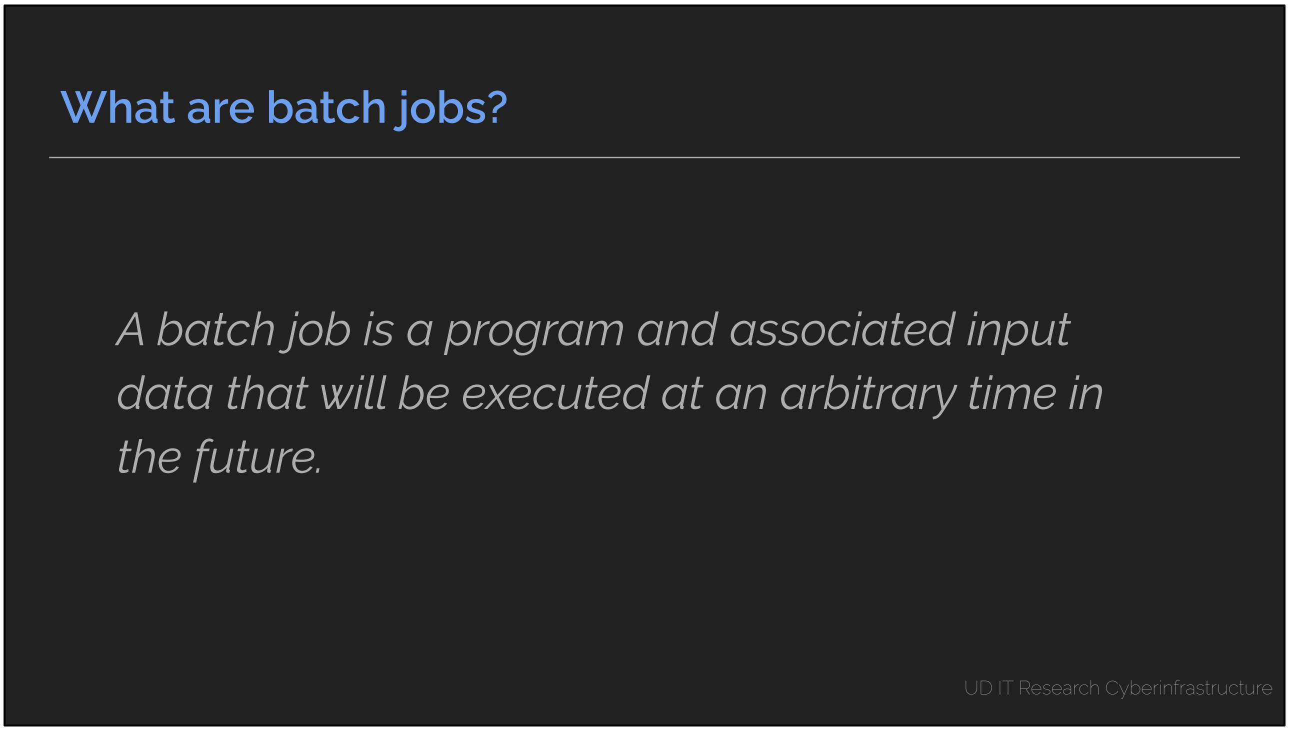
```
[frey@login01.darwin ~]$ ssqueue --format="%all" --state=PD --sort=-p,i --start

 ACCOUNT       | TRES_PER_NODE | MIN_CPUS | MIN_TMP_DISK | END_TIME            | FEATURES | GROUP        | OVER_SUBSCRIBE |   JOBID | NAME
 ──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
 mgi           | N/A           |        1 | 24G          | 2024-09-18T11:26:27 | (null)   | mgi          | OK             | 5306877 | msta
 bluecrabs     | N/A           |       50 | 0            | 2024-09-14T18:11:37 | (null)   | bluecrabs    | OK             | 5306423 | open
 xg-che220076  | N/A           |       64 | 0            | 2024-09-12T23:31:42 | (null)   | xg-che220076 | OK             | 5305158 | vasp
 xg-che220076  | N/A           |       64 | 0            | 2024-09-12T23:31:42 | (null)   | xg-che220076 | OK             | 5305159 | vasp
 xg-che220076  | N/A           |       64 | 0            | 2024-09-12T23:31:42 | (null)   | xg-che220076 | OK             | 5305160 | vasp
 xg-che220076  | N/A           |       64 | 0            | 2024-09-12T23:31:42 | (null)   | xg-che220076 | OK             | 5305161 | vasp
 xg-che220076  | N/A           |       64 | 0            | 2024-09-15T14:44:31 | (null)   | xg-che220076 | OK             | 5300707 | vasp
 xg-che220076  | N/A           |       64 | 0            | 2024-09-15T14:44:31 | (null)   | xg-che220076 | OK             | 5300708 | vasp
 xg-che220076  | N/A           |       64 | 0            | 2024-09-15T14:44:31 | (null)   | xg-che220076 | OK             | 5304804 | vasp
 xg-che220076  | N/A           |       64 | 0            | 2024-09-15T14:44:31 | (null)   | xg-che220076 | OK             | 5305035 | vasp
 safrono       | N/A           |        1 | 0            | N/A                 | (null)   | safrono      | USER           | 5304823 | ci+a
 safrono       | N/A           |        1 | 0            | N/A                 | (null)   | safrono      | USER           | 5304824 | ci+s
 mtg           | N/A           |       64 | 0            | N/A                 | (null)   | mtg          | OK             | 5303255 | bulk
 mtg           | N/A           |       64 | 0            | N/A                 | (null)   | mtg          | OK             | 5303256 | bulk
 ──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
  [Q]uit    [P]rev/[N]ext page    Page [L]eft/[R]ight    [E]nd/[B]eginning of list
```

Source code of ssqueue et al. available at: `https://gitlab.com/udel-itrci/slurm-output-wrappers.git`

May be hard to see the difference — the "squeue" has had an "s" prefixed on it
Many of the Slurm query commands can be prefixed with an "s" to get an interactive "spreadsheet" view.

What are jobs?

- A *job* is the computational work to be done — commands that would be typed at the shell prompt by the user
- When those commands are placed in a text file, a *shell script* has been created: the basis for a *batch job*
    - Shell scripting skills are **very** useful in crafting and submitting jobs
    - IT RCI is offering a separate Shell Scripting workshop on Oct 23



What are batch jobs?

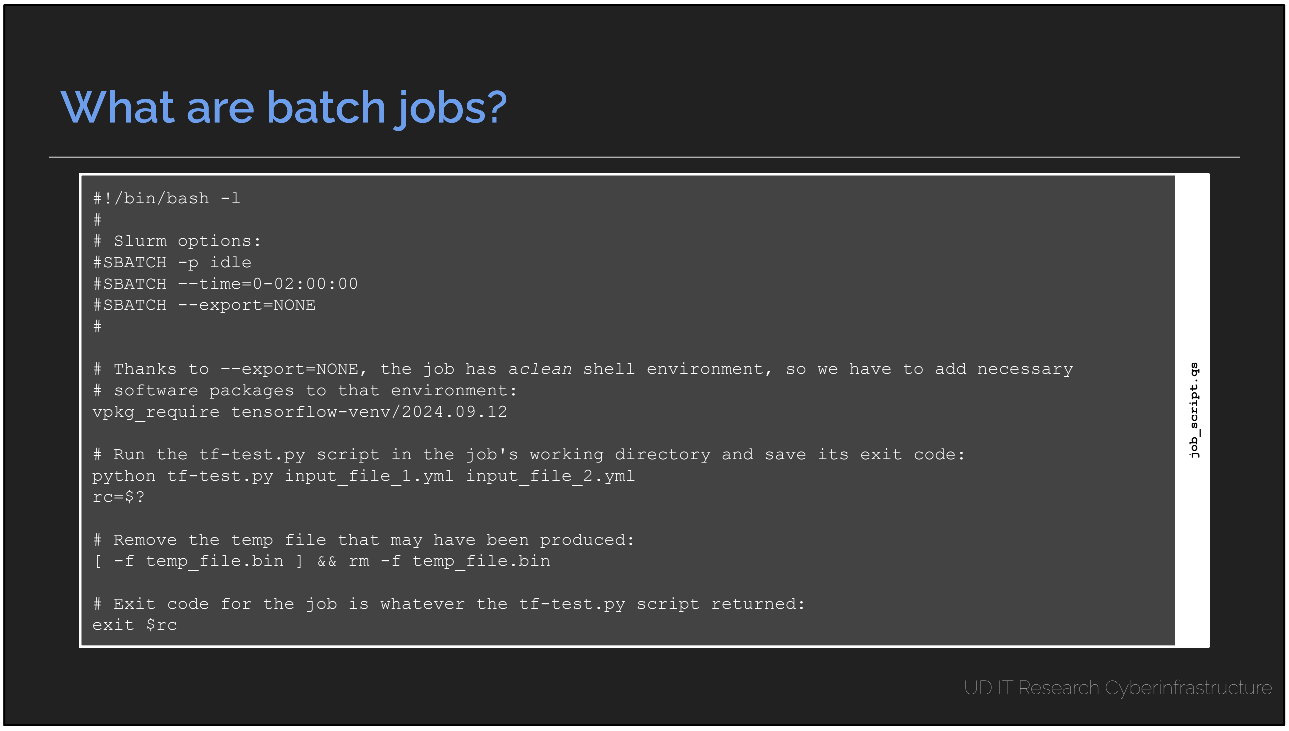
A batch job is a program and associated input data that will be executed at an arbitrary time in the future.

UD IT Research Cyberinfrastructure

Matlab is most often used as an interactive program: a GUI is presented, the user types and clicks to perform tasks. Matlab can also process a sequence of commands captured in a file, sometimes without any user interaction. The latter is Matlab operating in batch mode.

What are batch jobs?

- The program is a shell script, a.k.a. the *batch script*
- All inputs to the programs and commands in the batch script **must** come from arguments and files
  - Interaction with the user is **not** possible
- A script that runs without interactive inputs can be run at any time
  - This allows the job scheduler to schedule execution, reorder job priority, etc.
- Slurm retains a copy of the batch script when the job is submitted
  - The original script *can* be edited after the job is submitted…
  - …but all other files the job will use are **not** copied, so avoid modifying them!



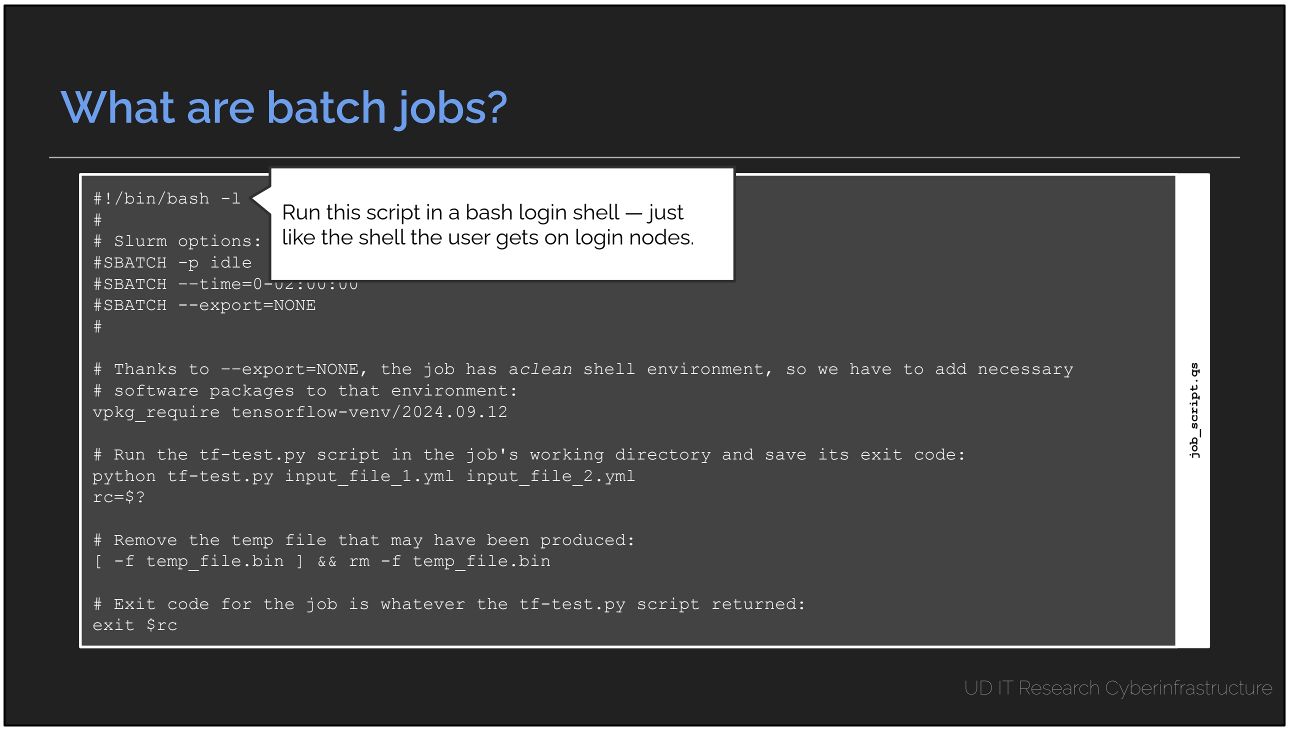
What are batch jobs?

job_script.qs

```
#!/bin/bash -l
#
# Slurm options:
#SBATCH -p idle
#SBATCH --time=0-02:00:00
#SBATCH --export=NONE
#

# Thanks to --export=NONE, the job has aclean shell environment, so we have to add necessary
# software packages to that environment:
vpkg_require tensorflow-venv/2024.09.12

# Run the tf-test.py script in the job's working directory and save its exit code:
python tf-test.py input_file_1.yml input_file_2.yml
rc=$?

# Remove the temp file that may have been produced:
[ -f temp_file.bin ] && rm -f temp_file.bin

# Exit code for the job is whatever the tf-test.py script returned:
exit $rc
```

UD IT Research Cyberinfrastructure

IT RCI prefers to use the extension ".qs" on batch scripts to differentiate them from regular shell scripts



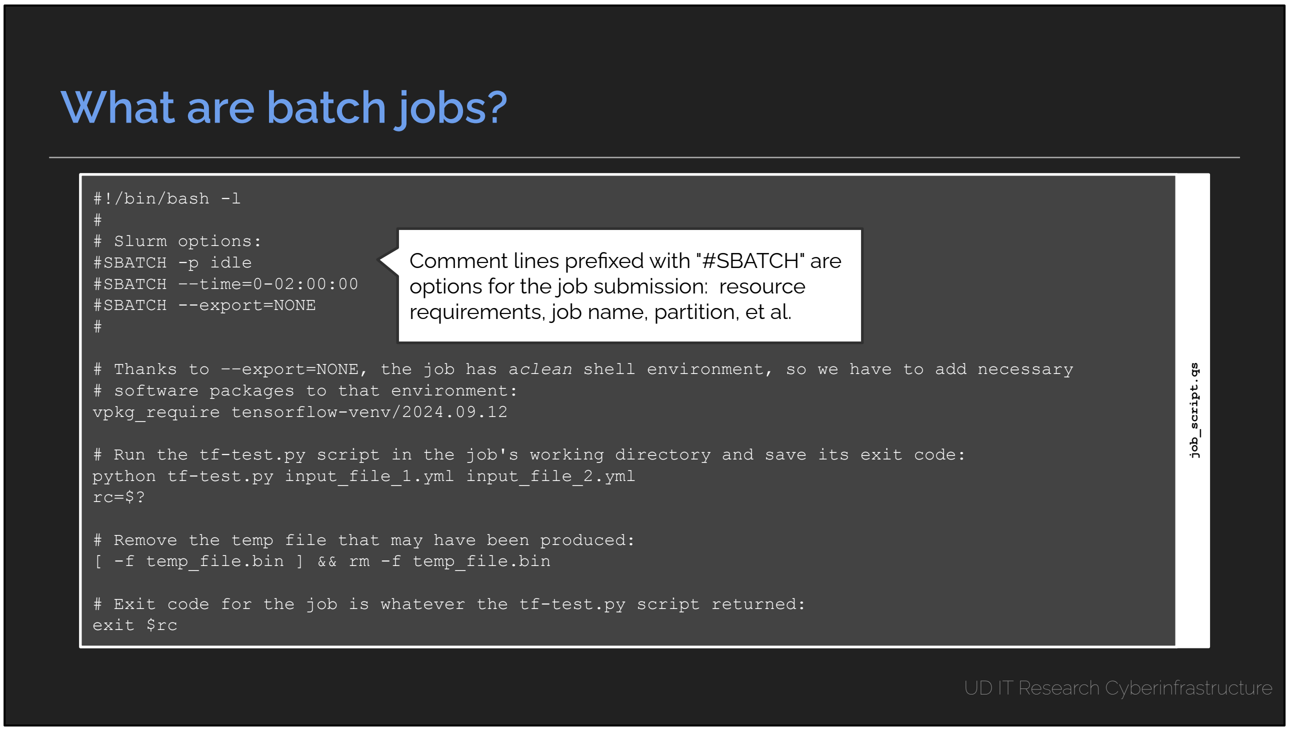
What are batch jobs?

```
#!/bin/bash -l
#
# Slurm options:
#SBATCH -p idle
#SBATCH --time=0-02:00:00
#SBATCH --export=NONE
#

# Thanks to --export=NONE, the job has aclean shell environment, so we have to add necessary
# software packages to that environment:
vpkg_require tensorflow-venv/2024.09.12

# Run the tf-test.py script in the job's working directory and save its exit code:
python tf-test.py input_file_1.yml input_file_2.yml
rc=$?

# Remove the temp file that may have been produced:
[ -f temp_file.bin ] && rm -f temp_file.bin

# Exit code for the job is whatever the tf-test.py script returned:
exit $rc
```

Run this script in a bash login shell — just like the shell the user gets on login nodes.

job_script.qs

Without the "-l" the environment will not have VALET or Slurm available to the batch script.



What are batch jobs?

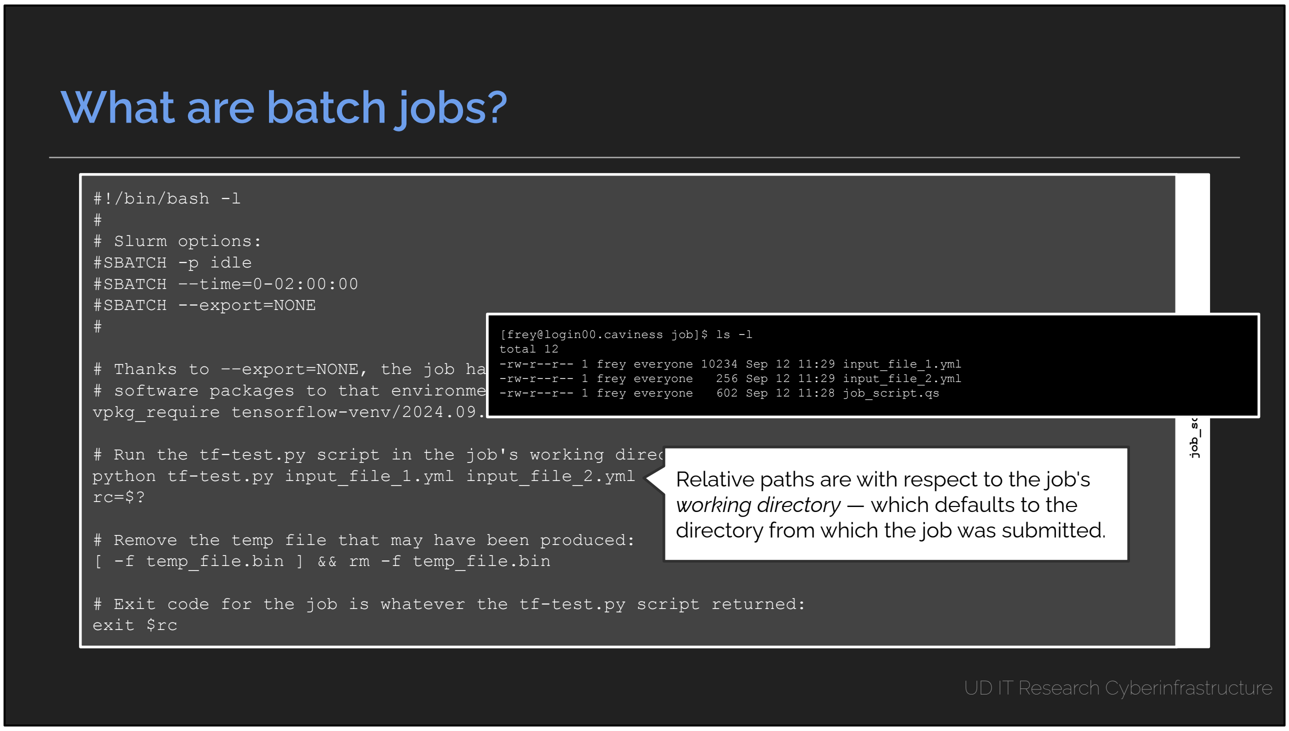
```
#!/bin/bash -l
#
# Slurm options:
#SBATCH -p idle
#SBATCH --time=0-02:00:00
#SBATCH --export=NONE
#

# Thanks to --export=NONE, the job has aclean shell environment, so we have to add necessary
# software packages to that environment:
vpkg_require tensorflow-venv/2024.09.12

# Run the tf-test.py script in the job's working directory and save its exit code:
python tf-test.py input_file_1.yml input_file_2.yml
rc=$?

# Remove the temp file that may have been produced:
[ -f temp_file.bin ] && rm -f temp_file.bin

# Exit code for the job is whatever the tf-test.py script returned:
exit $rc
```

job_script.qs

Comment lines prefixed with "#SBATCH" are options for the job submission: resource requirements, job name, partition, et al.

UD IT Research Cyberinfrastructure

First occurrence of anything other than a comment line ends parsing of options!!
The --export=NONE is recommended strongly by IT RCI so the job's shell environment starts from the same state as the login shells.



What are batch jobs?

```
#!/bin/bash -l
#
# Slurm options:
#SBATCH -p idle
#SBATCH --time=0-02:00:00
#SBATCH --export=NONE
#

# Thanks to --export=NONE, the job ha
# software packages to that environme
vpkg_require tensorflow-venv/2024.09.

# Run the tf-test.py script in the job's working direc
python tf-test.py input_file_1.yml input_file_2.yml
rc=$?

# Remove the temp file that may have been produced:
[ -f temp_file.bin ] && rm -f temp_file.bin

# Exit code for the job is whatever the tf-test.py script returned:
exit $rc
```

```
[frey@login00.caviness job]$ ls -l
total 12
-rw-r--r-- 1 frey everyone 10234 Sep 12 11:29 input_file_1.yml
-rw-r--r-- 1 frey everyone   256 Sep 12 11:29 input_file_2.yml
-rw-r--r-- 1 frey everyone   602 Sep 12 11:28 job_script.qs
```

Relative paths are with respect to the job's *working directory* — which defaults to the directory from which the job was submitted.

UD IT Research Cyberinfrastructure

A basic tenet of organizing computational work is to associate a job with a directory that will be that job's working directory; multiple jobs that work on data in sequence can reuse the same directory.

What are batch jobs?

```
#!/bin/bash -l
#
# Slurm options:
#SBATCH -p idle
#SBATCH --time=0-02:00:00
#SBATCH --export=NONE
#

# Thanks to --export=NONE, the job has aclean shell environment, so we have to add necessary
# software packages to that environment:
vpkg_require tensorflow-venv/2024.09.12

# Run the tf-test.py script in the job's working directory and save its exit code:
python t
rc=$?

# Remove
[ -f tem

# Exit code for the job is whatever the tf-test.py script returned:
exit $rc
```

job_script.qs

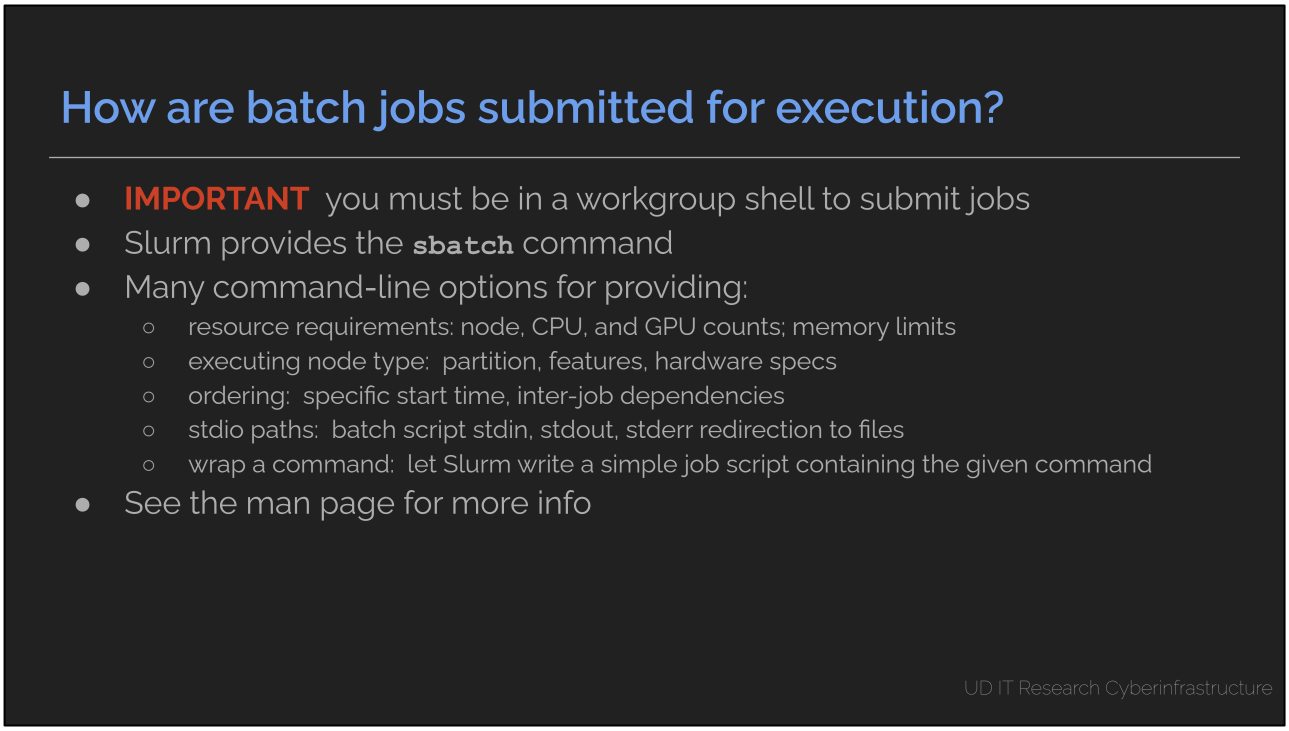
The $? in bash is the return code from the previous command: `0` implies success, non-zero values are error codes.

What are batch jobs?

job_script.qs

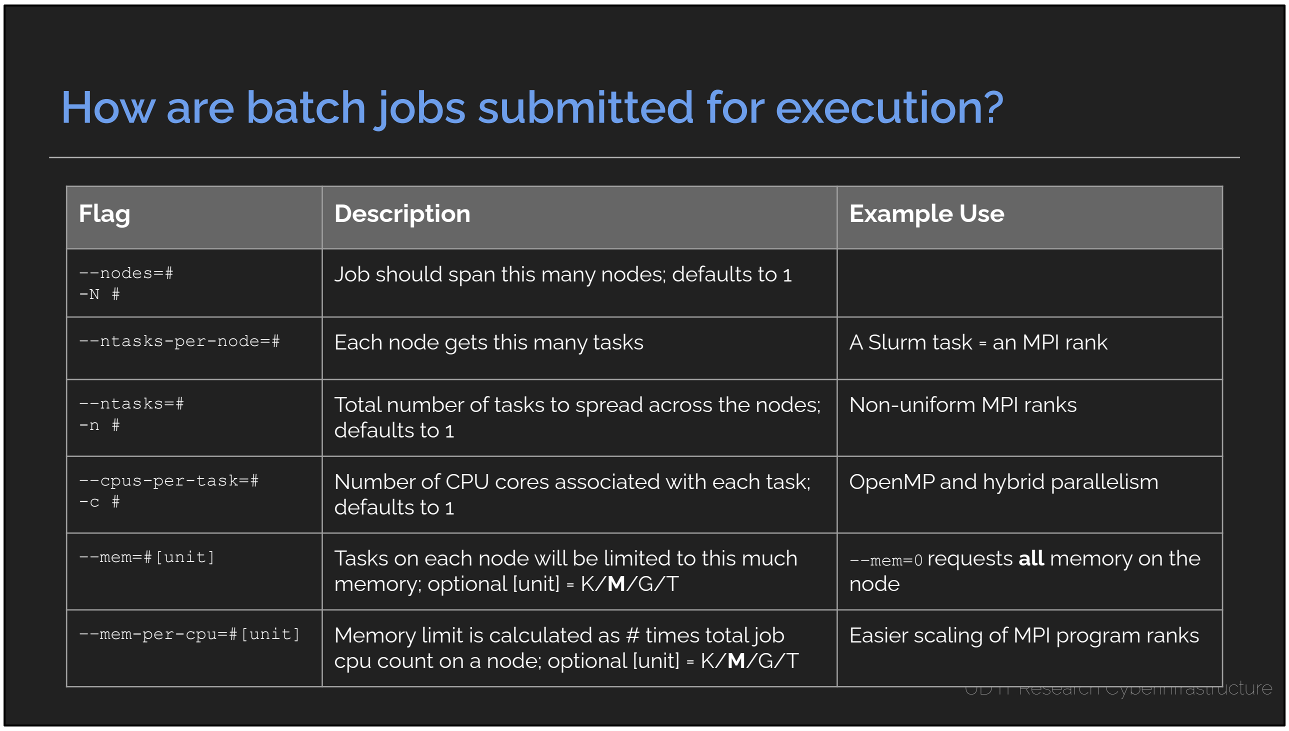
```
#!/bin/bash -l
#
# Slurm options:
#SBATCH -p idle
#SBATCH --time=0-02:00:00
#SBATCH --export=NONE
#

# Thanks to --export=NONE, the job has aclean shell environment, so we have to add necessary
# software packages to that environment:
vpkg_require tensorflow-venv/2024.09.12

# Run the tf-test.py script in the job's working directory and save its exit code:
python tf-test.py input_file_1.yml input_file_2.yml
rc=$?

# Remove the temp file that may have been produced:
[ -f temp_file.bin ] && rm -f temp_file.bin

# Exit code                                      pt returned:
exit $rc
```

The exit code from the batch script becomes the job's exit status: `0` = success, not `0` = failure.



How are batch jobs submitted for execution?

- **IMPORTANT** you must be in a workgroup shell to submit jobs
- Slurm provides the **`sbatch`** command
- Many command-line options for providing:
  - resource requirements: node, CPU, and GPU counts; memory limits
  - executing node type: partition, features, hardware specs
  - ordering: specific start time, inter-job dependencies
  - stdio paths: batch script stdin, stdout, stderr redirection to files
  - wrap a command: let Slurm write a simple job script containing the given command
- See the man page for more info



How are batch jobs submitted for execution?

Flag	Description	Example Use

`--nodes=#`
`-N #`
	Job should span this many nodes; defaults to 1	

`--ntasks-per-node=#`	Each node gets this many tasks	A Slurm task = an MPI rank

`--ntasks=#`
`-n #`
	Total number of tasks to spread across the nodes; defaults to 1	Non-uniform MPI ranks

`--cpus-per-task=#`
`-c #`
	Number of CPU cores associated with each task; defaults to 1	OpenMP and hybrid parallelism

`--mem=#[unit]`	Tasks on each node will be limited to this much memory; optional [unit] = K/**M**/G/T	`--mem=0` requests **all** memory on the node
`--mem-per-cpu=#[unit]`	Memory limit is calculated as # times total job cpu count on a node; optional [unit] = K/**M**/G/T	Easier scaling of MPI program ranks

Omitting the [unit] in memory specifications implies megabyte

How are batch jobs submitted for execution?

- Time is a consumable resource, too!

Flag	Description	Example Use
`--begin=<time>`	Job should not start until after this date/time	Delay for data to be ready for d/l
`--deadline=<time>`	Remove job is it cannot complete before this date/time	If job won't be completed by date, don't run it at all
`--time=<time>`	Job will run no longer than this duration (wall time limit); defaults to 30 minutes	Necessary on all jobs
`--time-min=<time>`	Provide a lower bound to the wall time limit, making `--time` an upper bound	Useful for backfill (flexible wall time makes for easier fit)

How are batch jobs submitted for execution?

- Time is a consumable resource, too!
  - Dates/times/durations in various forms

<time>	Description
`HH:MM{:SS}`	As a time, the given time today or tomorrow (if already passed) Also acts as a duration
`D-HH:MM{:SS}`	Duration including *D* 24-hour periods
`YYYY-MM-DD{THH:MM{:SS}}`	Specific date and time; midnight is implied if time is omitted
`today | tomorrow`	Midnight this day or the next
`midnight | noon | fika | teatime`	Specific time of day today or tomorrow; "fika" is Swedish coffee break (3 p.m.) and "teatime" is English tea (4 p.m.)
`now+<offset>`	Current date and time plus an offset (in min, hr, day, week)

How are batch jobs submitted for execution?

- GPUs are requested differently on Caviness vs. DARWIN
    - Caviness currently uses an older version of Slurm:

Flag	Description
`--gres=gpu:<type>`	Requests one GPU of <type> per node
`--gres=gpu:<type>:#`	Requests # GPUs of <type> per node
`<type> = p100 | v100 | t4 | a100 | a40`	

How are batch jobs submitted for execution?

- GPUs are requested differently on Caviness vs. DARWIN
  - Slurm on DARWIN has tighter integration and detection of GPUs

Flag	Description
`--gpus=<type>`	Requests one GPU of <type> per node
`--gpus=<type>:#`	Requests # GPUs of <type> per node
`<type> = tesla_t4 | tesla_v100 | amd_mi50 | amd_mi100`	

How are batch jobs submitted for execution?

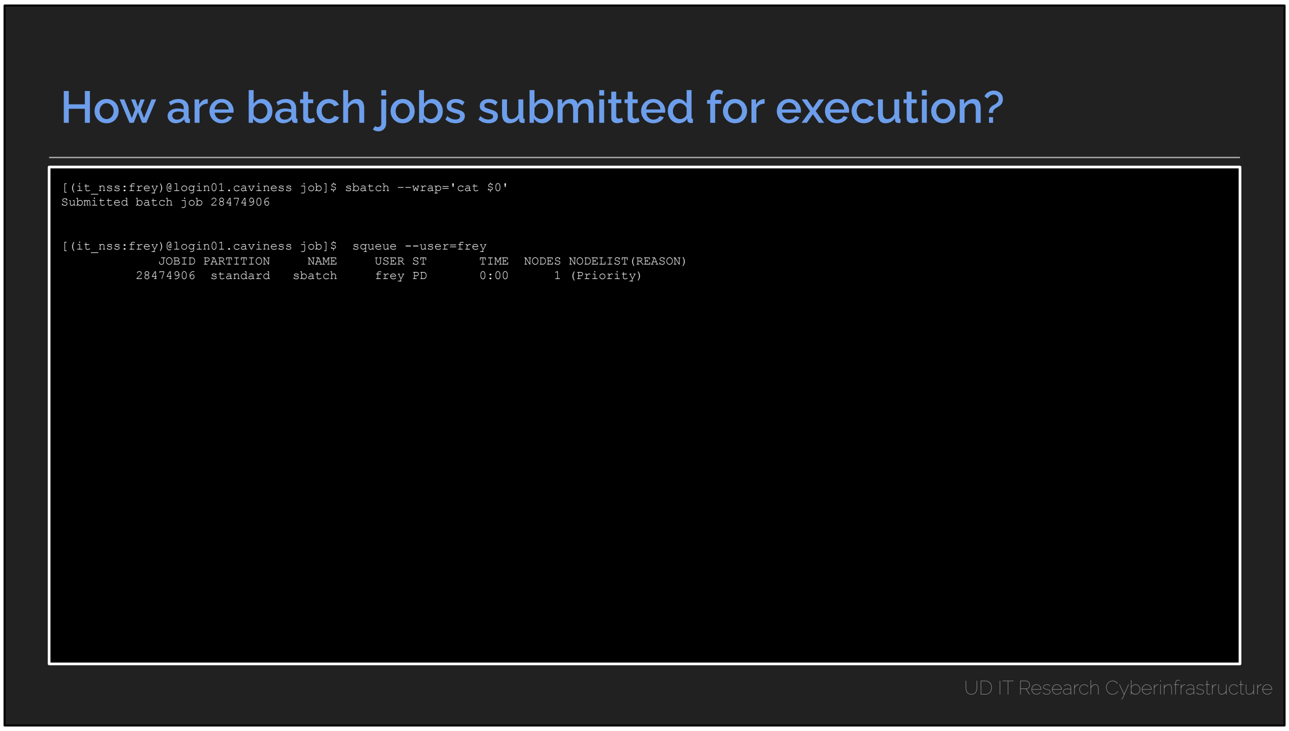
- GPUs are requested differently on Caviness vs. DARWIN
- On both clusters device cgroups are used to limit GPU access
    - A job requesting 2 of 4 GPUs on the node has 2 specific devices assigned to it
    - The other 2 GPUs are **not visible** to the job
    - Enumerating devices (e.g. `nvidia-smi`) will show two devices
        - Indices will be 0 and 1, regardless of absolute index of device
            - Physical GPUs 2 and 4 assigned, indices will be 0 and 1
        - Empty CUDA_VISIBLE_DEVICES implies both devices

How are batch jobs submitted for execution?

- It is the user's responsibility to match resource quantities to jobs
  - Over-requesting CPUs, GPUs, or memory
    - User and workgroup will be billed for unused resources
      - Debit to allocation on DARWIN
      - Decreased priority via unnecessarily-high usage history
    - Idle resources could be employed by other cluster users
  - Inflated time limit
    - E.g. request 7-day time limit for job that actually needs much shorter period
    - Impacts the accuracy, optimality of the job schedule
    - Impacts user's own jobs — harder to find large blocks of time in schedule

How are batch jobs submitted for execution?

- It is the user's responsibility to match resource quantities to jobs
- *Benchmarking* is critical to this responsibility
    - With any new computational software or method, run a set of representative problems
    - Observe peak memory usage versus computational parameters
        - Memory usage will usually vary based on "problem size"
    - For parallel programs, rerun the same problem while varying CPU count
        - E.g. 1, 2, 4, 8, 16 CPUs on a single node, MPI spanning 1, 2, etc. nodes
        - Analyze *scaling* of the problem: how efficiently do additional CPUs cut wall time?
    - The sacct command is helpful (stay tuned)
    - Every job run is another data point in benchmarking
    - Workgroups would do well to aggregate, document, and share internally
        - Easy for existing — and future — group members to consistently make better choices



How are batch jobs submitted for execution?

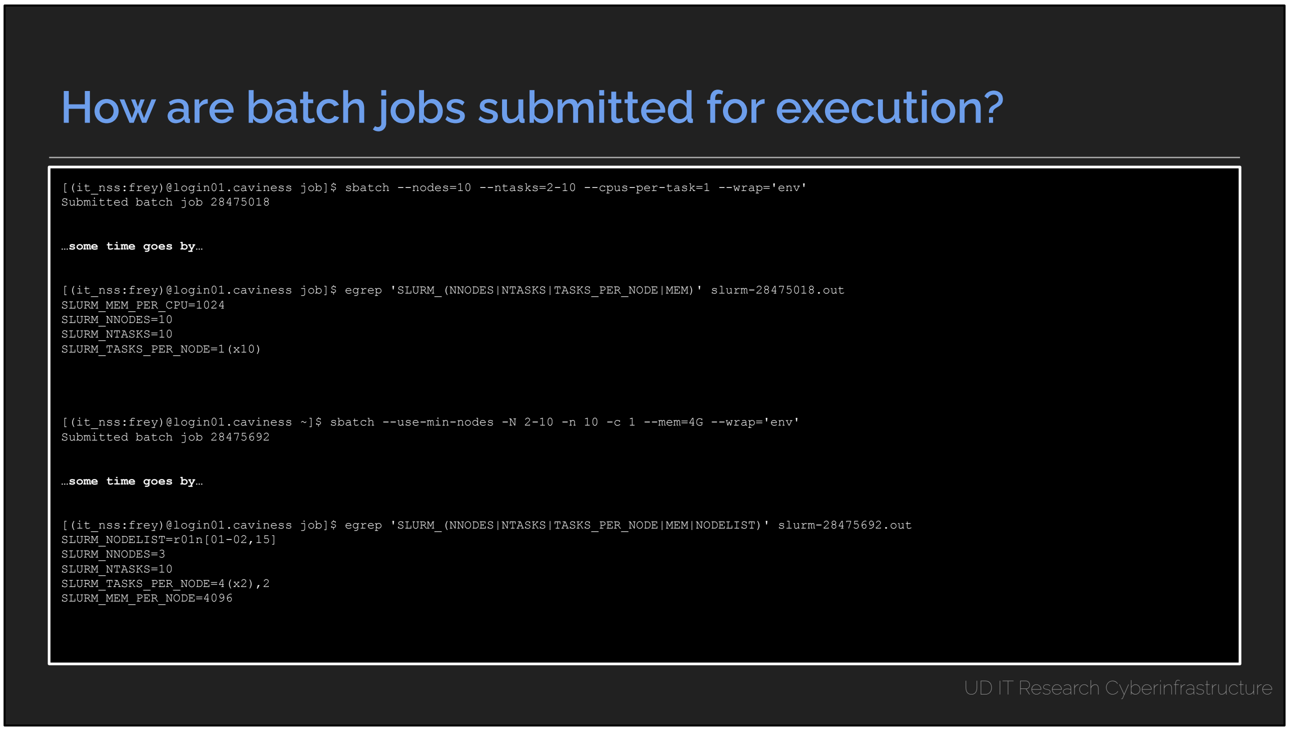
```
[(it_nss:frey)@login01.caviness job]$ sbatch --wrap='cat $0'
Submitted batch job 28474906


[(it_nss:frey)@login01.caviness job]$  squeue --user=frey
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          28474906  standard   sbatch     frey PD       0:00      1 (Priority)
```

UD IT Research Cyberinfrastructure

We will return to squeue later

How are batch jobs submitted for execution?

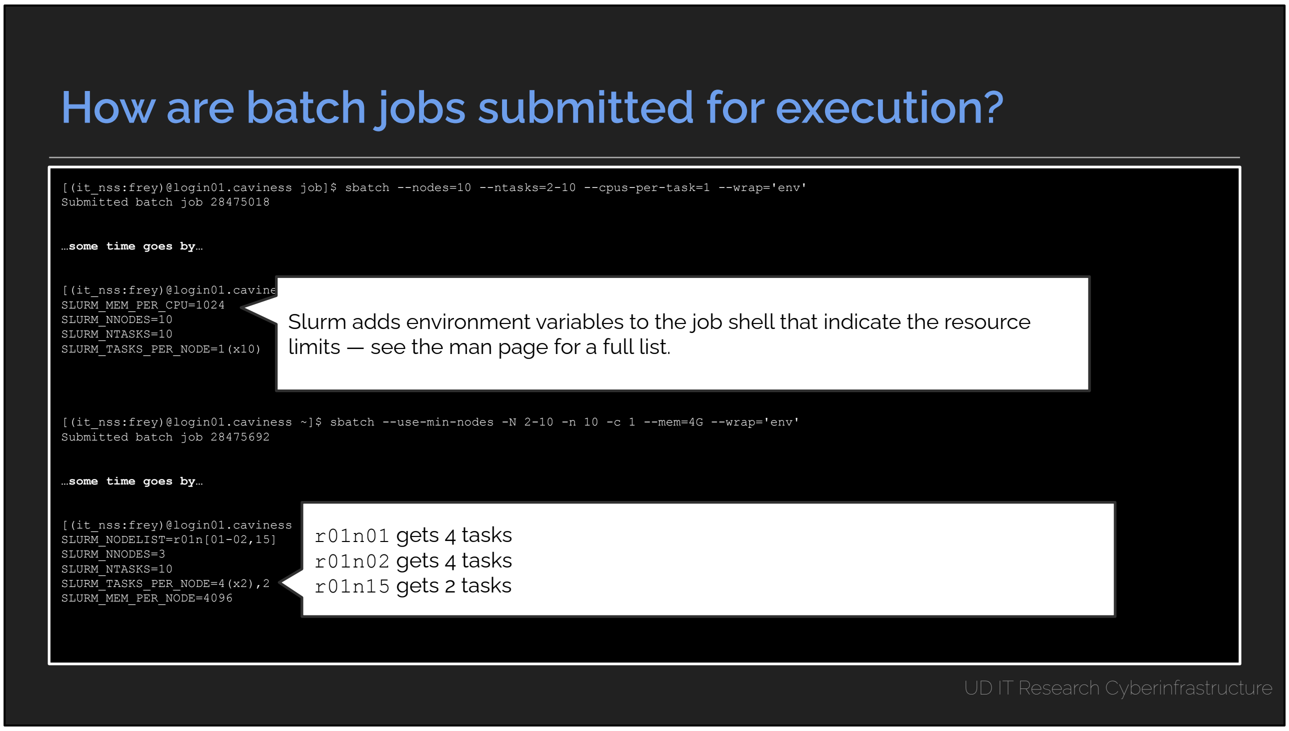
```
[(it_nss:frey)@login01.caviness job]$ sbatch --wrap='cat $0'
Submitted batch job 28474906


[(it_nss:frey)@login01.caviness job]$  squeue --user=frey
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          28474906  standard   sbatch     frey PD       0:00      1 (Priority)


…some time goes by…


[(it_nss:frey)@login01.caviness job]$ cat slurm-28474906.out
#!/bin/sh
# This script was created by sbatch --wrap.

cat $0
```



How are batch jobs submitted for execution?

```
[(it_nss:frey)@login01.caviness job]$ sbatch --nodes=10 --ntasks=2-10 --cpus-per-task=1 --wrap='env'
Submitted batch job 28475018


…some time goes by…


[(it_nss:frey)@login01.caviness job]$ egrep 'SLURM_(NNODES|NTASKS|TASKS_PER_NODE|MEM)' slurm-28475018.out
SLURM_MEM_PER_CPU=1024
SLURM_NNODES=10
SLURM_NTASKS=10
SLURM_TASKS_PER_NODE=1(x10)



[(it_nss:frey)@login01.caviness ~]$ sbatch --use-min-nodes -N 2-10 -n 10 -c 1 --mem=4G --wrap='env'
Submitted batch job 28475692


…some time goes by…


[(it_nss:frey)@login01.caviness job]$ egrep 'SLURM_(NNODES|NTASKS|TASKS_PER_NODE|MEM|NODELIST)' slurm-28475692.out
SLURM_NODELIST=r01n[01-02,15]
SLURM_NNODES=3
SLURM_NTASKS=10
SLURM_TASKS_PER_NODE=4(x2),2
SLURM_MEM_PER_NODE=4096
```

Slurm loves to abbreviate lists that contain repetitive items — which often makes variable values harder to work with. E.g. SLURM_NODELIST and SLURM_TASKS_PER_NODE.



How are batch jobs submitted for execution?

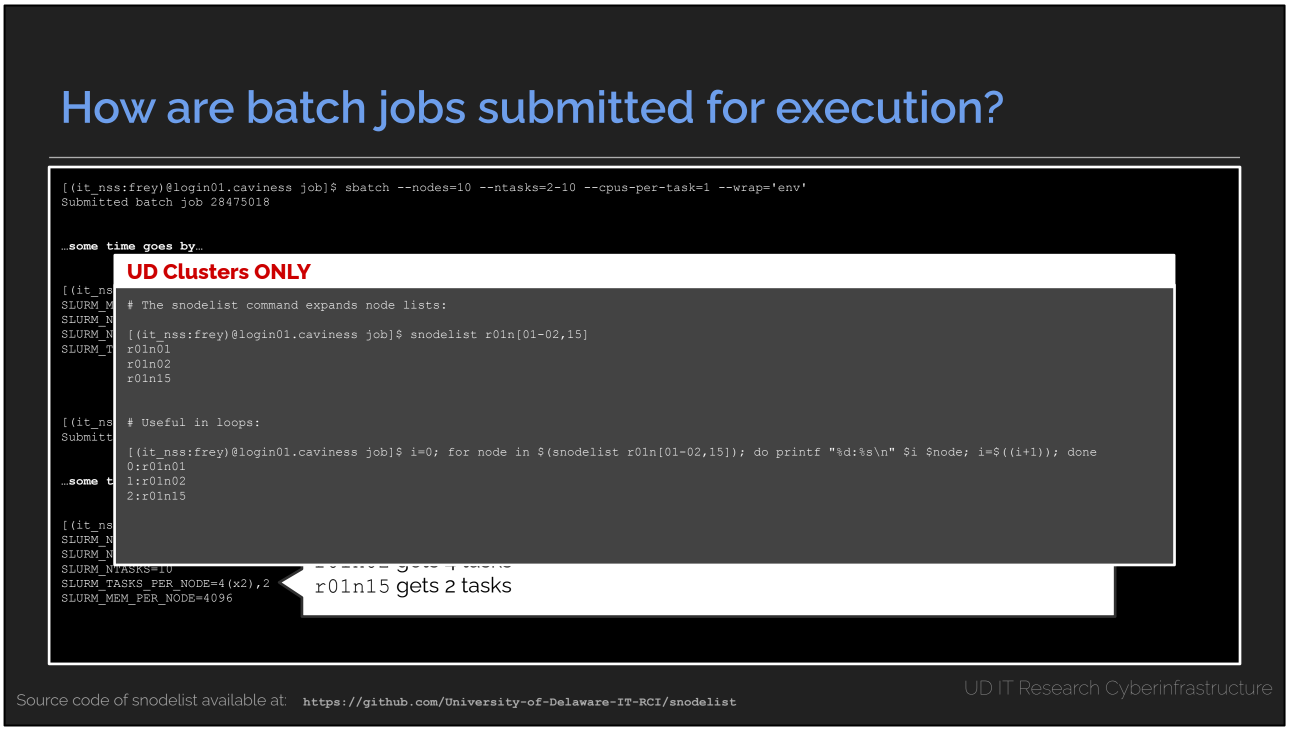
```
[(it_nss:frey)@login01.caviness job]$ sbatch --nodes=10 --ntasks=2-10 --cpus-per-task=1 --wrap='env'
Submitted batch job 28475018


…some time goes by…


[(it_nss:frey)@login01.cavine
SLURM_MEM_PER_CPU=1024
SLURM_NNODES=10
SLURM_NTASKS=10
SLURM_TASKS_PER_NODE=1(x10)
```

Slurm adds environment variables to the job shell that indicate the resource limits — see the man page for a full list.

```
[(it_nss:frey)@login01.caviness ~]$ sbatch --use-min-nodes -N 2-10 -n 10 -c 1 --mem=4G --wrap='env'
Submitted batch job 28475692


…some time goes by…


[(it_nss:frey)@login01.caviness
SLURM_NODELIST=r01n[01-02,15]
SLURM_NNODES=3
SLURM_NTASKS=10
SLURM_TASKS_PER_NODE=4(x2),2
SLURM_MEM_PER_NODE=4096
```

`r01n01` gets 4 tasks
`r01n02` gets 4 tasks
`r01n15` gets 2 tasks

Slurm loves to abbreviate lists that contain repetitive items — which often makes variable values harder to work with. E.g. SLURM_NODELIST and SLURM_TASKS_PER_NODE.



How are batch jobs submitted for execution?

```
[(it_nss:frey)@login01.caviness job]$ sbatch --nodes=10 --ntasks=2-10 --cpus-per-task=1 --wrap='env'
Submitted batch job 28475018

…some time goes by…

[(it_ns
SLURM_M
SLURM_N
SLURM_N
SLURM_T

[(it_ns
Submitt

…some t

[(it_ns
SLURM_N
SLURM_N
SLURM_NTASKS=10
SLURM_TASKS_PER_NODE=4(x2),2
SLURM_MEM_PER_NODE=4096
```

r01n15 gets 2 tasks

UD Clusters ONLY

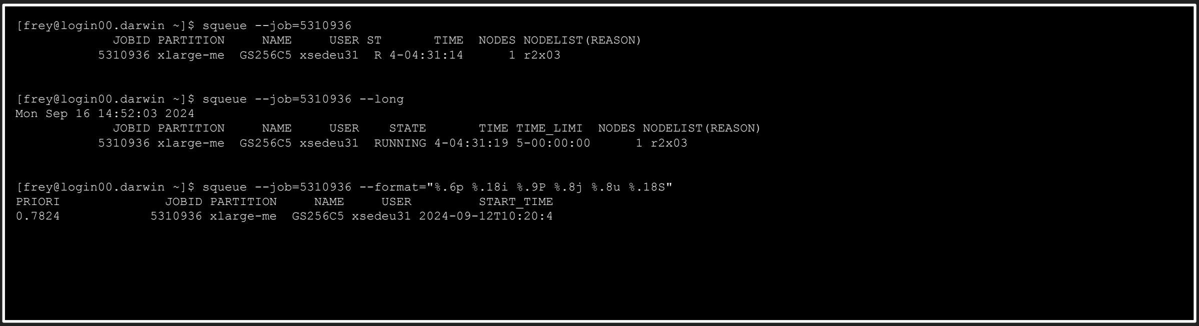
```
# The snodelist command expands node lists:

[(it_nss:frey)@login01.caviness job]$ snodelist r01n[01-02,15]
r01n01
r01n02
r01n15


# Useful in loops:

[(it_nss:frey)@login01.caviness job]$ i=0; for node in $(snodelist r01n[01-02,15]); do printf "%d:%s\n" $i $node; i=$((i+1)); done
0:r01n01
1:r01n02
2:r01n15
```

Source code of snodelist available at: `https://github.com/University-of-Delaware-IT-RCI/snodelist`

Slurm loves to abbreviate lists that contain repetitive items — which often makes variable values harder to work with. E.g. SLURM_NODELIST and SLURM_TASKS_PER_NODE.

How are batch jobs monitored?

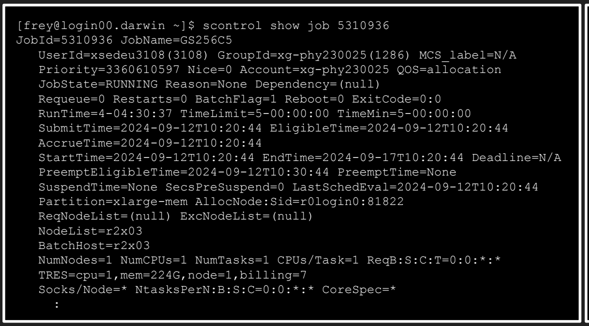
- When submitted, the *job id* is displayed
  - The *job id* is a unique integer that identifies an individual job moving through Slurm
- Monitoring active jobs:
  - The squeue command can again be used, with the ––job=# flag



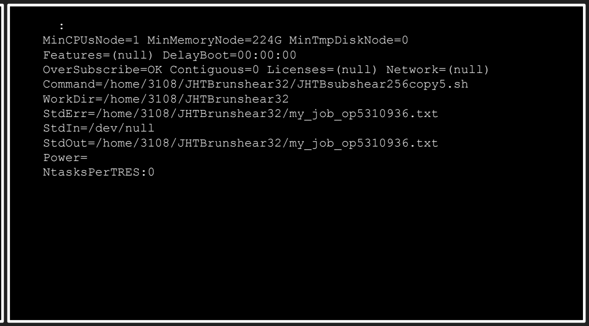
```
[frey@login00.darwin ~]$ squeue --job=5310936
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           5310936 xlarge-me  GS256C5 xsedeu31  R 4-04:31:14      1 r2x03


[frey@login00.darwin ~]$ squeue --job=5310936 --long
Mon Sep 16 14:52:03 2024
             JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
           5310936 xlarge-me  GS256C5 xsedeu31  RUNNING 4-04:31:19 5-00:00:00      1 r2x03


[frey@login00.darwin ~]$ squeue --job=5310936 --format="%.6p %.18i %.9P %.8j %.8u %.18S"
PRIORI              JOBID PARTITION     NAME     USER         START_TIME
0.7824            5310936 xlarge-me  GS256C5 xsedeu31 2024-09-12T10:20:4
```

How are batch jobs monitored?

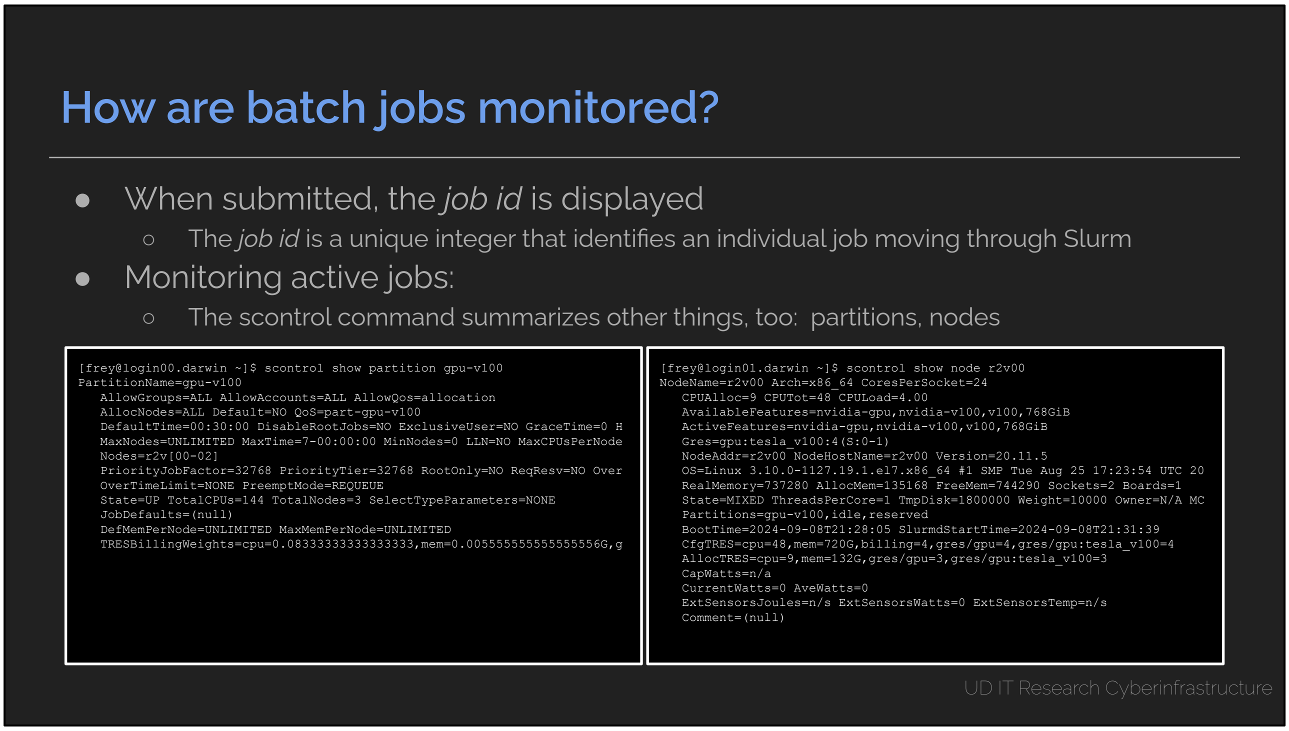
- When submitted, the *job id* is displayed
  - The *job id* is a unique integer that identifies an individual job moving through Slurm
- Monitoring active jobs:
  - The scontrol command summarizes the intrinsic job data



```
[frey@login00.darwin ~]$ scontrol show job 5310936
JobId=5310936 JobName=GS256C5
   UserId=xsedeu3108(3108) GroupId=xg-phy230025(1286) MCS_label=N/A
   Priority=3360610597 Nice=0 Account=xg-phy230025 QOS=allocation
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=0 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=4-04:30:37 TimeLimit=5-00:00:00 TimeMin=5-00:00:00
   SubmitTime=2024-09-12T10:20:44 EligibleTime=2024-09-12T10:20:44
   AccrueTime=2024-09-12T10:20:44
   StartTime=2024-09-12T10:20:44 EndTime=2024-09-17T10:20:44 Deadline=N/A
   PreemptEligibleTime=2024-09-12T10:30:44 PreemptTime=None
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2024-09-12T10:20:44
   Partition=xlarge-mem AllocNode:Sid=r0login0:81822
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=r2x03
   BatchHost=r2x03
   NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=1,mem=224G,node=1,billing=7
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
     :
```



```
     :
   MinCPUsNode=1 MinMemoryNode=224G MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/home/3108/JHTBrunshear32/JHTBsubshear256copy5.sh
   WorkDir=/home/3108/JHTBrunshear32
   StdErr=/home/3108/JHTBrunshear32/my_job_op5310936.txt
   StdIn=/dev/null
   StdOut=/home/3108/JHTBrunshear32/my_job_op5310936.txt
   Power=
   NtasksPerTRES:0
```



How are batch jobs monitored?

- When submitted, the *job id* is displayed
  - The *job id* is a unique integer that identifies an individual job moving through Slurm
- Monitoring active jobs:
  - The scontrol command summarizes other things, too: partitions, nodes

```
[frey@login00.darwin ~]$ scontrol show partition gpu-v100
PartitionName=gpu-v100
   AllowGroups=ALL AllowAccounts=ALL AllowQos=allocation
   AllocNodes=ALL Default=NO QoS=part-gpu-v100
   DefaultTime=00:30:00 DisableRootJobs=NO ExclusiveUser=NO GraceTime=0 H
   MaxNodes=UNLIMITED MaxTime=7-00:00:00 MinNodes=0 LLN=NO MaxCPUsPerNode
   Nodes=r2v[00-02]
   PriorityJobFactor=32768 PriorityTier=32768 RootOnly=NO ReqResv=NO Over
   OverTimeLimit=NONE PreemptMode=REQUEUE
   State=UP TotalCPUs=144 TotalNodes=3 SelectTypeParameters=NONE
   JobDefaults=(null)
   DefMemPerNode=UNLIMITED MaxMemPerNode=UNLIMITED
   TRESBillingWeights=cpu=0.08333333333333333,mem=0.005555555555555556G,g
```

```
[frey@login01.darwin ~]$ scontrol show node r2v00
NodeName=r2v00 Arch=x86_64 CoresPerSocket=24
   CPUAlloc=9 CPUTot=48 CPULoad=4.00
   AvailableFeatures=nvidia-gpu,nvidia-v100,v100,768GiB
   ActiveFeatures=nvidia-gpu,nvidia-v100,v100,768GiB
   Gres=gpu:tesla_v100:4(S:0-1)
   NodeAddr=r2v00 NodeHostName=r2v00 Version=20.11.5
   OS=Linux 3.10.0-1127.19.1.el7.x86_64 #1 SMP Tue Aug 25 17:23:54 UTC 20
   RealMemory=737280 AllocMem=135168 FreeMem=744290 Sockets=2 Boards=1
   State=MIXED ThreadsPerCore=1 TmpDisk=1800000 Weight=10000 Owner=N/A MC
   Partitions=gpu-v100,idle,reserved
   BootTime=2024-09-08T21:28:05 SlurmdStartTime=2024-09-08T21:31:39
   CfgTRES=cpu=48,mem=720G,billing=4,gres/gpu=4,gres/gpu:tesla_v100=4
   AllocTRES=cpu=9,mem=132G,gres/gpu=3,gres/gpu:tesla_v100=3
   CapWatts=n/a
   CurrentWatts=0 AveWatts=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
   Comment=(null)
```

If you omit the partition or node name, all of the item are listed

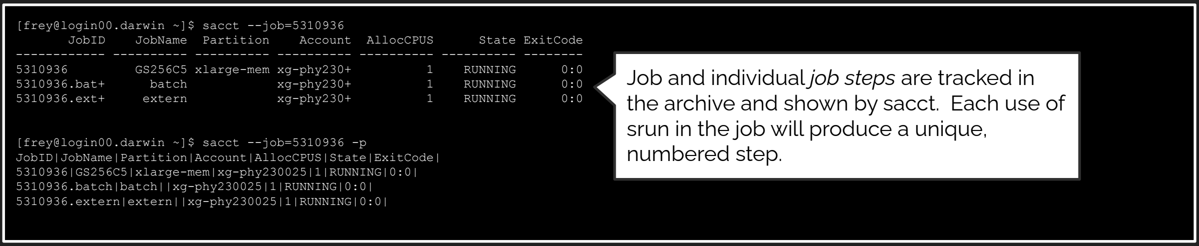
How are batch jobs monitored?

- When submitted, the *job id* is displayed
  - The *job id* is a unique integer that identifies an individual job moving through Slurm
- Monitoring past and active jobs:
  - Slurm configured to archive job information into a database
    - Necessary for tracking usage history (and calculating job priorities)
  - Job information is queried using the sacct command

```
[frey@login00.darwin ~]$ sacct --job=5310936
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
5310936         GS256C5 xlarge-mem xg-phy230+          1    RUNNING      0:0
5310936.bat+      batch            xg-phy230+          1    RUNNING      0:0
5310936.ext+     extern            xg-phy230+          1    RUNNING      0:0


[frey@login00.darwin ~]$ sacct --job=5310936 -p
JobID|JobName|Partition|Account|AllocCPUS|State|ExitCode|
5310936|GS256C5|xlarge-mem|xg-phy230025|1|RUNNING|0:0|
5310936.batch|batch||xg-phy230025|1|RUNNING|0:0|
5310936.extern|extern||xg-phy230025|1|RUNNING|0:0|
```

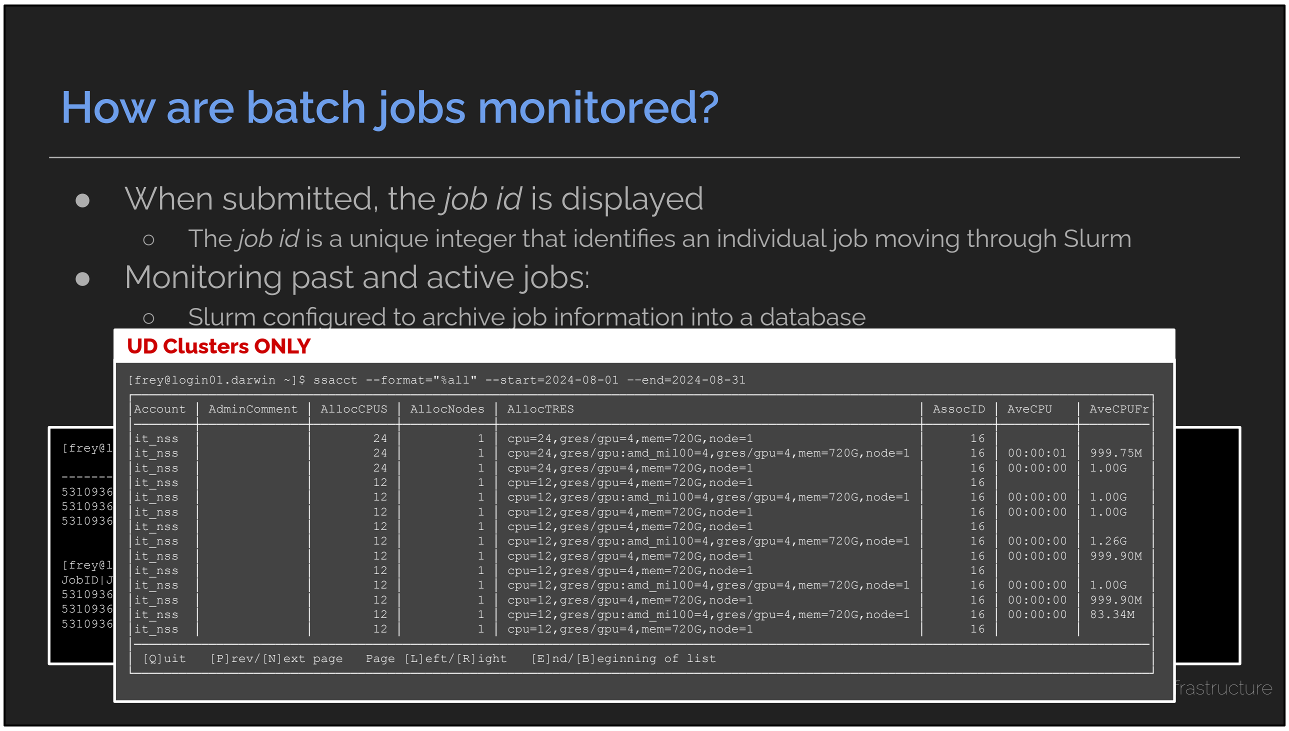
How are batch jobs monitored?

- When submitted, the *job id* is displayed
    - The *job id* is a unique integer that identifies an individual job moving through Slurm
- Monitoring past and active jobs:
    - Slurm configured to archive job information into a database
        - Necessary for tracking usage history (and calculating job priorities)
    - Job information is queried using the sacct command



```
[frey@login00.darwin ~]$ sacct --job=5310936
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
5310936         GS256C5 xlarge-mem xg-phy230+          1    RUNNING      0:0
5310936.bat+      batch            xg-phy230+          1    RUNNING      0:0
5310936.ext+     extern            xg-phy230+          1    RUNNING      0:0


[frey@login00.darwin ~]$ sacct --job=5310936 -p
JobID|JobName|Partition|Account|AllocCPUS|State|ExitCode|
5310936|GS256C5|xlarge-mem|xg-phy230025|1|RUNNING|0:0|
5310936.batch|batch||xg-phy230025|1|RUNNING|0:0|
5310936.extern|extern||xg-phy230025|1|RUNNING|0:0|
```

Job and individual *job steps* are tracked in the archive and shown by sacct. Each use of srun in the job will produce a unique, numbered step.



How are batch jobs monitored?

- When submitted, the *job id* is displayed
  - The *job id* is a unique integer that identifies an individual job moving through Slurm
- Monitoring past and active jobs:
  - Slurm configured to archive job information into a database

```
[frey@l

-------
5310936
5310936
5310936


[frey@l
JobID|J
5310936
5310936
5310936
```

UD Clusters ONLY

```
[frey@login01.darwin ~]$ ssacct --format="%all" --start=2024-08-01 --end=2024-08-31

 Account | AdminComment | AllocCPUS | AllocNodes | AllocTRES                                                   | AssocID | AveCPU   | AveCPUFr
---------------------------------------------------------------------------------------------------------------------------------------------------
 it_nss  |              |        24 |          1 | cpu=24,gres/gpu=4,mem=720G,node=1                           |      16 |          |
 it_nss  |              |        24 |          1 | cpu=24,gres/gpu:amd_mi100=4,gres/gpu=4,mem=720G,node=1      |      16 | 00:00:01 | 999.75M
 it_nss  |              |        24 |          1 | cpu=24,gres/gpu=4,mem=720G,node=1                           |      16 | 00:00:00 | 1.00G
 it_nss  |              |        12 |          1 | cpu=12,gres/gpu=4,mem=720G,node=1                           |      16 |          |
 it_nss  |              |        12 |          1 | cpu=12,gres/gpu:amd_mi100=4,gres/gpu=4,mem=720G,node=1      |      16 | 00:00:00 | 1.00G
 it_nss  |              |        12 |          1 | cpu=12,gres/gpu=4,mem=720G,node=1                           |      16 | 00:00:00 | 1.00G
 it_nss  |              |        12 |          1 | cpu=12,gres/gpu=4,mem=720G,node=1                           |      16 |          |
 it_nss  |              |        12 |          1 | cpu=12,gres/gpu:amd_mi100=4,gres/gpu=4,mem=720G,node=1      |      16 | 00:00:00 | 1.26G
 it_nss  |              |        12 |          1 | cpu=12,gres/gpu=4,mem=720G,node=1                           |      16 | 00:00:00 | 999.90M
 it_nss  |              |        12 |          1 | cpu=12,gres/gpu=4,mem=720G,node=1                           |      16 |          |
 it_nss  |              |        12 |          1 | cpu=12,gres/gpu:amd_mi100=4,gres/gpu=4,mem=720G,node=1      |      16 | 00:00:00 | 1.00G
 it_nss  |              |        12 |          1 | cpu=12,gres/gpu=4,mem=720G,node=1                           |      16 | 00:00:00 | 999.90M
 it_nss  |              |        12 |          1 | cpu=12,gres/gpu:amd_mi100=4,gres/gpu=4,mem=720G,node=1      |      16 | 00:00:00 | 83.34M
 it_nss  |              |        12 |          1 | cpu=12,gres/gpu=4,mem=720G,node=1                           |      16 |          |
---------------------------------------------------------------------------------------------------------------------------------------------------
  [Q]uit    [P]rev/[N]ext page    Page [L]eft/[R]ight    [E]nd/[B]eginning of list
```

frastructure

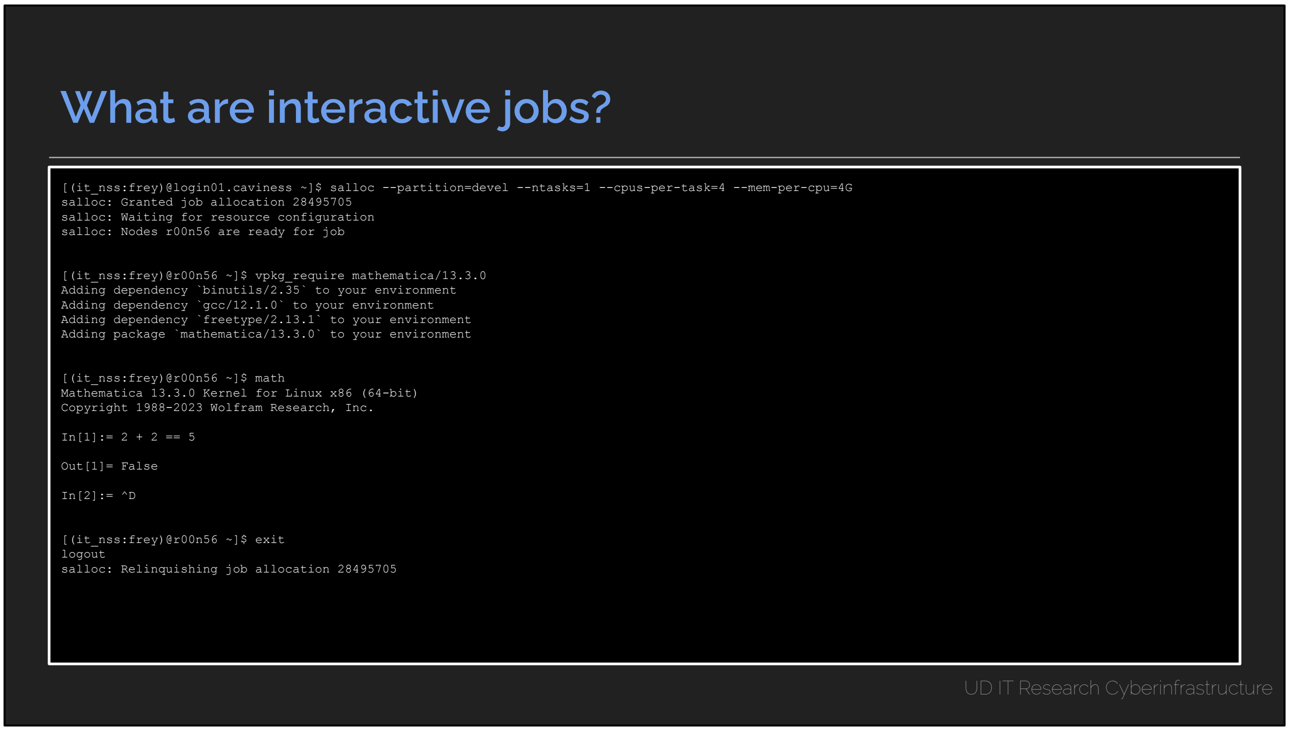
Just as with squeue, the "%all" field name implies all fields should be included in the output. See the sacct man page for a description of all fields.

What are interactive jobs?

- A batch job executes without user input at some future time
- Jobs that require user input are *interactive jobs*
  - Primary remote process for the job is a Bash shell
  - Typical use cases:
    - Benchmarking
    - Jupyter GUIs (web browser with SSH tunneling)
    - Matlab, Mathematica GUIs
    - Active debugging (e.g. gdb)

What are interactive jobs?

- A batch job executes without user input at some future time
- Jobs that require user input are *interactive jobs*
- **BE AWARE** interactive jobs consume resources *even when they are doing nothing*
  - An interactive job ties-up its requested CPU, GPU, and memory resources
  - The user (and the workgroup) are billed for the duration of the interactive job
    - NOT just actual computational time
  - An idle interactive job may be inconveniencing many cluster users

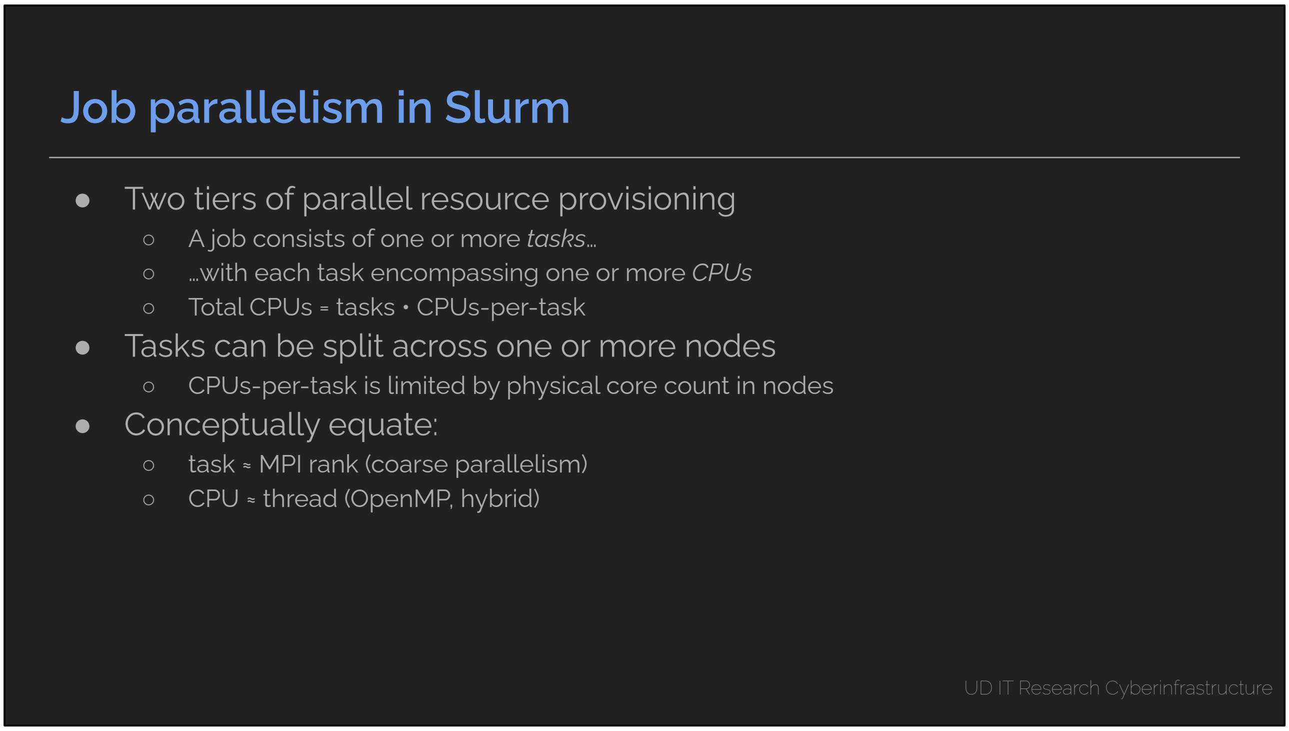


What are interactive jobs?

```
[(it_nss:frey)@login01.caviness ~]$ salloc --partition=devel --ntasks=1 --cpus-per-task=4 --mem-per-cpu=4G
salloc: Granted job allocation 28495705
salloc: Waiting for resource configuration
salloc: Nodes r00n56 are ready for job


[(it_nss:frey)@r00n56 ~]$ vpkg_require mathematica/13.3.0
Adding dependency `binutils/2.35` to your environment
Adding dependency `gcc/12.1.0` to your environment
Adding dependency `freetype/2.13.1` to your environment
Adding package `mathematica/13.3.0` to your environment


[(it_nss:frey)@r00n56 ~]$ math
Mathematica 13.3.0 Kernel for Linux x86 (64-bit)
Copyright 1988-2023 Wolfram Research, Inc.

In[1]:= 2 + 2 == 5

Out[1]= False

In[2]:= ^D


[(it_nss:frey)@r00n56 ~]$ exit
logout
salloc: Relinquishing job allocation 28495705
```

UD IT Research Cyberinfrastructure

The same options available to sbatch can be used with salloc.

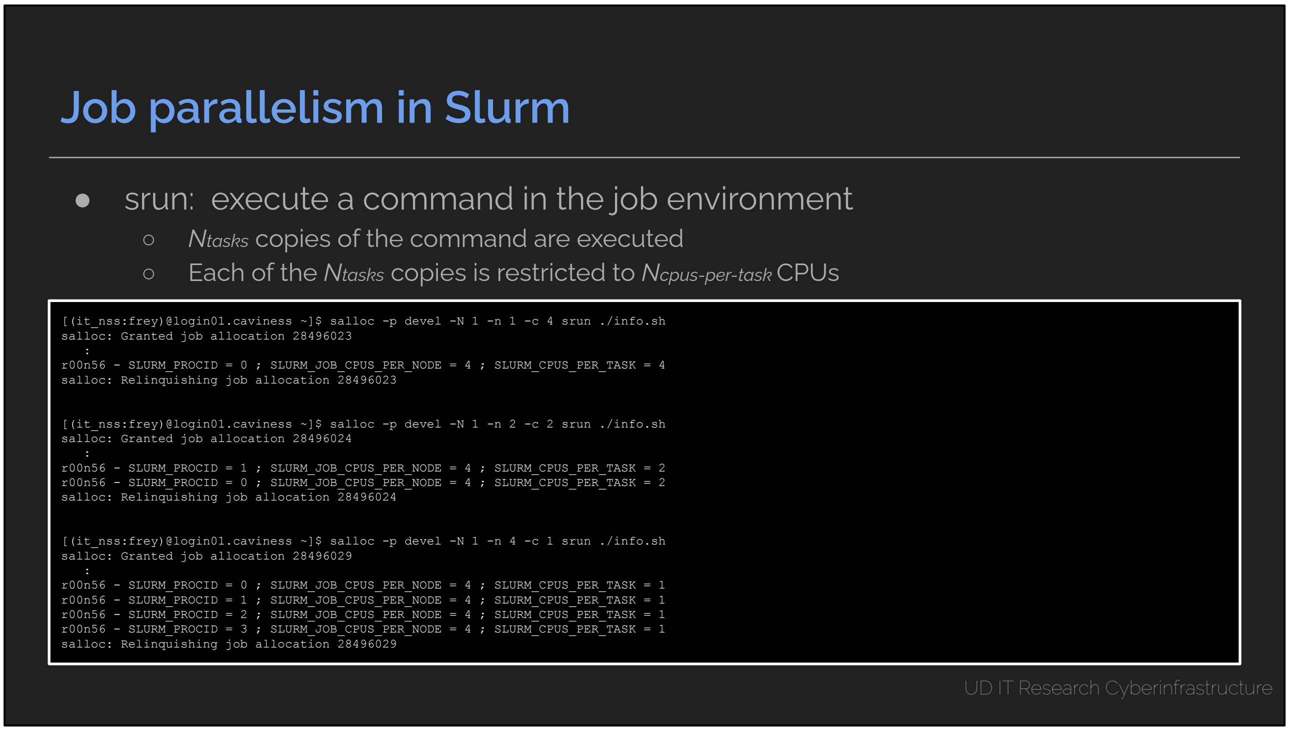


Job parallelism in Slurm

- Two tiers of parallel resource provisioning
  - A job consists of one or more *tasks*…
  - …with each task encompassing one or more *CPUs*
  - Total CPUs = tasks • CPUs-per-task
- Tasks can be split across one or more nodes
  - CPUs-per-task is limited by physical core count in nodes
- Conceptually equate:
  - task ≈ MPI rank (coarse parallelism)
  - CPU ≈ thread (OpenMP, hybrid)

UD IT Research Cyberinfrastructure

These concepts are documented in the comment header of our job script templates



Job parallelism in Slurm

- srun: execute a command in the job environment
  - *N*N_{tasks} copies of the command are executed
  - Each of the N_{tasks} copies is restricted to $N_{cpus\text{-}per\text{-}task}$ CPUs

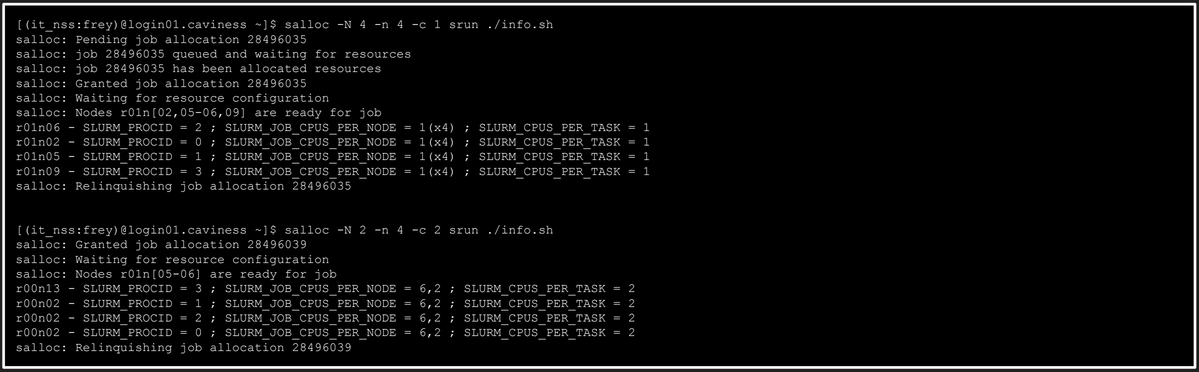
```
[(it_nss:frey)@login01.caviness ~]$ salloc -p devel -N 1 -n 1 -c 4 srun ./info.sh
salloc: Granted job allocation 28496023
    :
r00n56 - SLURM_PROCID = 0 ; SLURM_JOB_CPUS_PER_NODE = 4 ; SLURM_CPUS_PER_TASK = 4
salloc: Relinquishing job allocation 28496023


[(it_nss:frey)@login01.caviness ~]$ salloc -p devel -N 1 -n 2 -c 2 srun ./info.sh
salloc: Granted job allocation 28496024
    :
r00n56 - SLURM_PROCID = 1 ; SLURM_JOB_CPUS_PER_NODE = 4 ; SLURM_CPUS_PER_TASK = 2
r00n56 - SLURM_PROCID = 0 ; SLURM_JOB_CPUS_PER_NODE = 4 ; SLURM_CPUS_PER_TASK = 2
salloc: Relinquishing job allocation 28496024


[(it_nss:frey)@login01.caviness ~]$ salloc -p devel -N 1 -n 4 -c 1 srun ./info.sh
salloc: Granted job allocation 28496029
    :
r00n56 - SLURM_PROCID = 0 ; SLURM_JOB_CPUS_PER_NODE = 4 ; SLURM_CPUS_PER_TASK = 1
r00n56 - SLURM_PROCID = 1 ; SLURM_JOB_CPUS_PER_NODE = 4 ; SLURM_CPUS_PER_TASK = 1
r00n56 - SLURM_PROCID = 2 ; SLURM_JOB_CPUS_PER_NODE = 4 ; SLURM_CPUS_PER_TASK = 1
r00n56 - SLURM_PROCID = 3 ; SLURM_JOB_CPUS_PER_NODE = 4 ; SLURM_CPUS_PER_TASK = 1
salloc: Relinquishing job allocation 28496029
```

UD IT Research Cyberinfrastructure

- The info.sh script prints the hostname of the compute node followed by the value of several variables Slurm adds to the job's environment

Job parallelism in Slurm

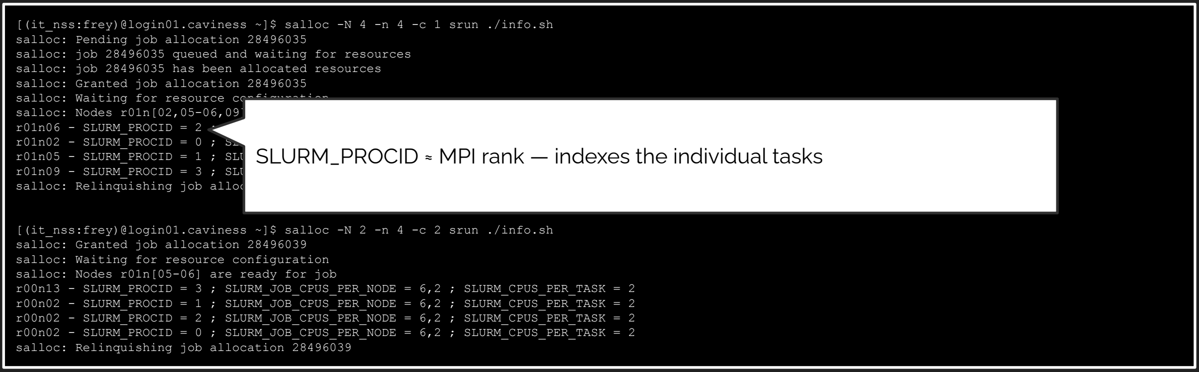
- srun: execute a command in the job environment
  - *N_{tasks}* copies of the command are executed
  - Each of the *N_{tasks}* copies is restricted to *N_{cpus-per-task}* CPUs



```
[(it_nss:frey)@login01.caviness ~]$ salloc -N 4 -n 4 -c 1 srun ./info.sh
salloc: Pending job allocation 28496035
salloc: job 28496035 queued and waiting for resources
salloc: job 28496035 has been allocated resources
salloc: Granted job allocation 28496035
salloc: Waiting for resource configuration
salloc: Nodes r01n[02,05-06,09] are ready for job
r01n06 - SLURM_PROCID = 2 ; SLURM_JOB_CPUS_PER_NODE = 1(x4) ; SLURM_CPUS_PER_TASK = 1
r01n02 - SLURM_PROCID = 0 ; SLURM_JOB_CPUS_PER_NODE = 1(x4) ; SLURM_CPUS_PER_TASK = 1
r01n05 - SLURM_PROCID = 1 ; SLURM_JOB_CPUS_PER_NODE = 1(x4) ; SLURM_CPUS_PER_TASK = 1
r01n09 - SLURM_PROCID = 3 ; SLURM_JOB_CPUS_PER_NODE = 1(x4) ; SLURM_CPUS_PER_TASK = 1
salloc: Relinquishing job allocation 28496035


[(it_nss:frey)@login01.caviness ~]$ salloc -N 2 -n 4 -c 2 srun ./info.sh
salloc: Granted job allocation 28496039
salloc: Waiting for resource configuration
salloc: Nodes r01n[05-06] are ready for job
r00n13 - SLURM_PROCID = 3 ; SLURM_JOB_CPUS_PER_NODE = 6,2 ; SLURM_CPUS_PER_TASK = 2
r00n02 - SLURM_PROCID = 1 ; SLURM_JOB_CPUS_PER_NODE = 6,2 ; SLURM_CPUS_PER_TASK = 2
r00n02 - SLURM_PROCID = 2 ; SLURM_JOB_CPUS_PER_NODE = 6,2 ; SLURM_CPUS_PER_TASK = 2
r00n02 - SLURM_PROCID = 0 ; SLURM_JOB_CPUS_PER_NODE = 6,2 ; SLURM_CPUS_PER_TASK = 2
salloc: Relinquishing job allocation 28496039
```

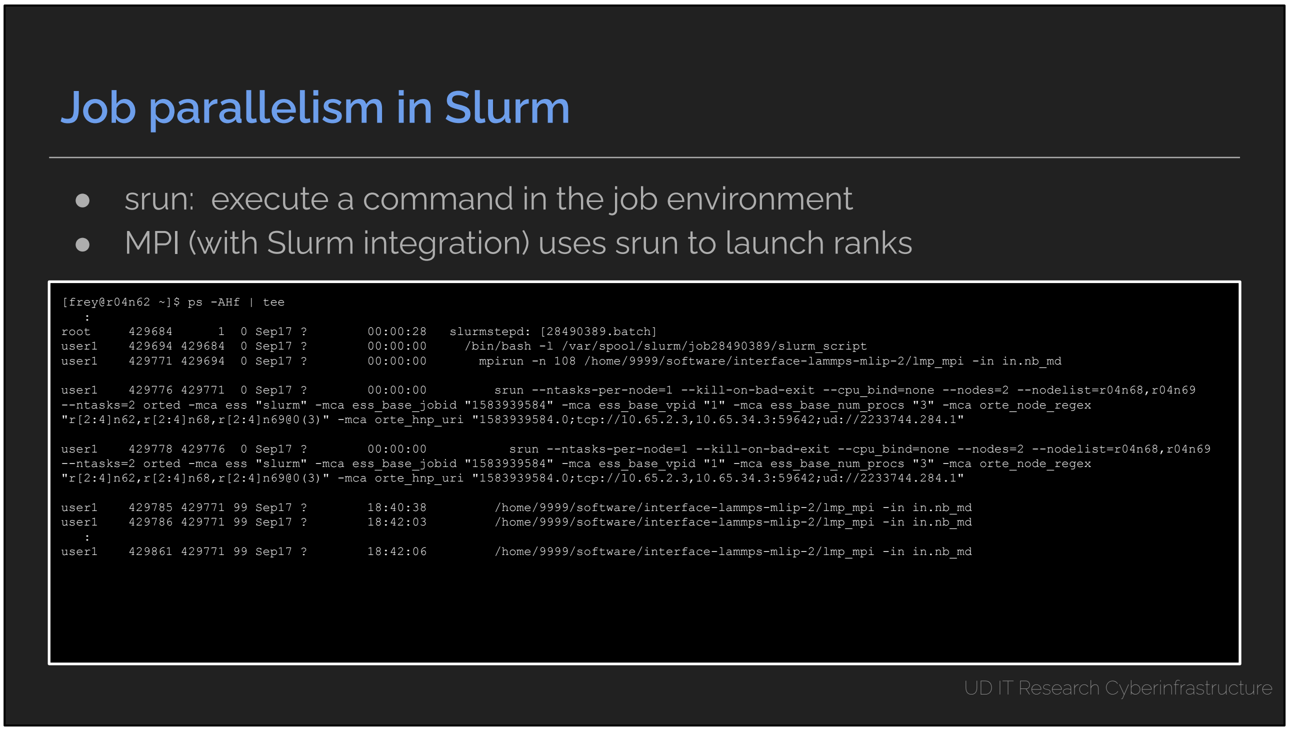
Job parallelism in Slurm

- srun: execute a command in the job environment
  - *N**tasks* copies of the command are executed
  - Each of the *Ntasks* copies is restricted to *Ncpus-per-task* CPUs



```
[(it_nss:frey)@login01.caviness ~]$ salloc -N 4 -n 4 -c 1 srun ./info.sh
salloc: Pending job allocation 28496035
salloc: job 28496035 queued and waiting for resources
salloc: job 28496035 has been allocated resources
salloc: Granted job allocation 28496035
salloc: Waiting for resource co
salloc: Nodes r01n[02,05-06,09]
r01n06 - SLURM_PROCID = 2
r01n02 - SLURM_PROCID = 0 ; S
r01n05 - SLURM_PROCID = 1 ; SLU
r01n09 - SLURM_PROCID = 3 ; SLU
salloc: Relinquishing job alloc


[(it_nss:frey)@login01.caviness ~]$ salloc -N 2 -n 4 -c 2 srun ./info.sh
salloc: Granted job allocation 28496039
salloc: Waiting for resource configuration
salloc: Nodes r01n[05-06] are ready for job
r00n13 - SLURM_PROCID = 3 ; SLURM_JOB_CPUS_PER_NODE = 6,2 ; SLURM_CPUS_PER_TASK = 2
r00n02 - SLURM_PROCID = 1 ; SLURM_JOB_CPUS_PER_NODE = 6,2 ; SLURM_CPUS_PER_TASK = 2
r00n02 - SLURM_PROCID = 2 ; SLURM_JOB_CPUS_PER_NODE = 6,2 ; SLURM_CPUS_PER_TASK = 2
r00n02 - SLURM_PROCID = 0 ; SLURM_JOB_CPUS_PER_NODE = 6,2 ; SLURM_CPUS_PER_TASK = 2
salloc: Relinquishing job allocation 28496039
```

SLURM_PROCID ≈ MPI rank — indexes the individual tasks

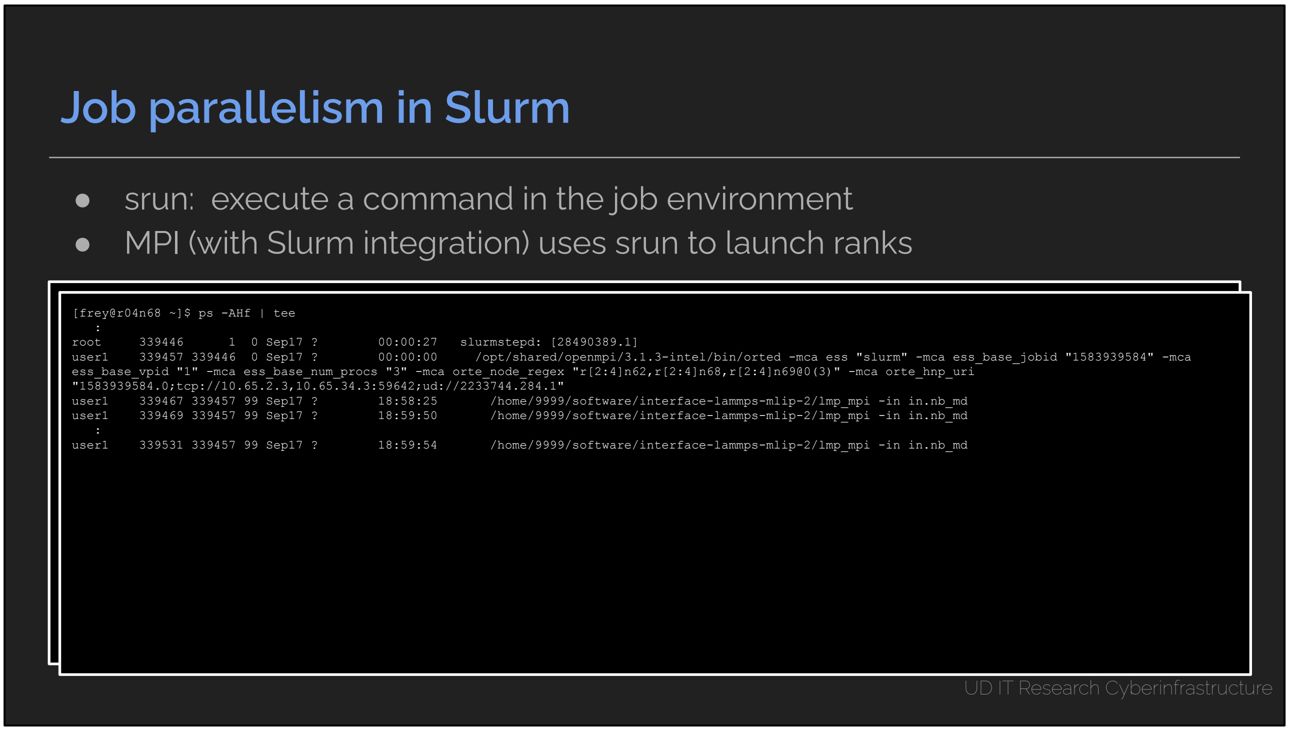


Job parallelism in Slurm

- srun: execute a command in the job environment
- MPI (with Slurm integration) uses srun to launch ranks

```
[frey@r04n62 ~]$ ps -AHf | tee
   :
root     429684      1  0 Sep17 ?        00:00:28   slurmstepd: [28490389.batch]
user1    429694 429684  0 Sep17 ?        00:00:00     /bin/bash -l /var/spool/slurm/job28490389/slurm_script
user1    429771 429694  0 Sep17 ?        00:00:00       mpirun -n 108 /home/9999/software/interface-lammps-mlip-2/lmp_mpi -in in.nb_md

user1    429776 429771  0 Sep17 ?        00:00:00         srun --ntasks-per-node=1 --kill-on-bad-exit --cpu_bind=none --nodes=2 --nodelist=r04n68,r04n69
--ntasks=2 orted -mca ess "slurm" -mca ess_base_jobid "1583939584" -mca ess_base_vpid "1" -mca ess_base_num_procs "3" -mca orte_node_regex
"r[2:4]n62,r[2:4]n68,r[2:4]n69@0(3)" -mca orte_hnp_uri "1583939584.0;tcp://10.65.2.3,10.65.34.3:59642;ud://2233744.284.1"

user1    429778 429776  0 Sep17 ?        00:00:00           srun --ntasks-per-node=1 --kill-on-bad-exit --cpu_bind=none --nodes=2 --nodelist=r04n68,r04n69
--ntasks=2 orted -mca ess "slurm" -mca ess_base_jobid "1583939584" -mca ess_base_vpid "1" -mca ess_base_num_procs "3" -mca orte_node_regex
"r[2:4]n62,r[2:4]n68,r[2:4]n69@0(3)" -mca orte_hnp_uri "1583939584.0;tcp://10.65.2.3,10.65.34.3:59642;ud://2233744.284.1"

user1    429785 429771 99 Sep17 ?        18:40:38         /home/9999/software/interface-lammps-mlip-2/lmp_mpi -in in.nb_md
user1    429786 429771 99 Sep17 ?        18:42:03         /home/9999/software/interface-lammps-mlip-2/lmp_mpi -in in.nb_md
   :
user1    429861 429771 99 Sep17 ?        18:42:06         /home/9999/software/interface-lammps-mlip-2/lmp_mpi -in in.nb_md
```

- The srun command OMITS the node on which the batch step is running; launches a single task on each remote node, the resulting orted process will spawn remote ranks



Job parallelism in Slurm

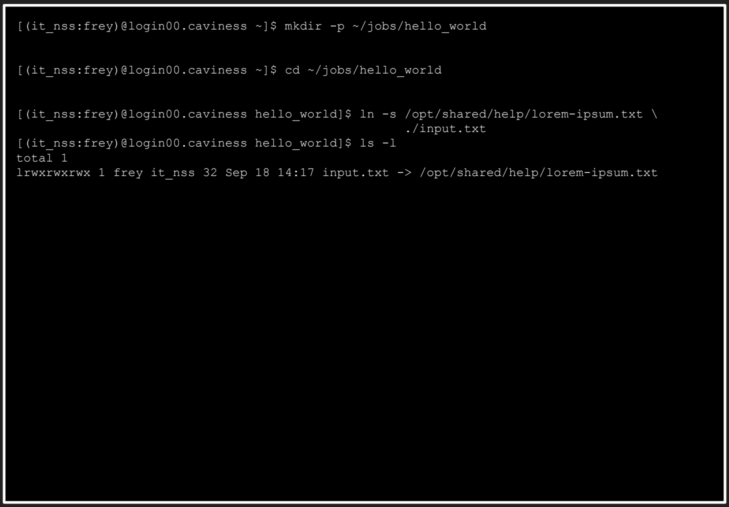
- srun: execute a command in the job environment
- MPI (with Slurm integration) uses srun to launch ranks

```
[frey@r04n68 ~]$ ps -AHf | tee
   :
root      339446      1  0 Sep17 ?        00:00:27   slurmstepd: [28490389.1]
user1     339457 339446  0 Sep17 ?        00:00:00     /opt/shared/openmpi/3.1.3-intel/bin/orted -mca ess "slurm" -mca ess_base_jobid "1583939584" -mca
ess_base_vpid "1" -mca ess_base_num_procs "3" -mca orte_node_regex "r[2:4]n62,r[2:4]n68,r[2:4]n69@0(3)" -mca orte_hnp_uri
"1583939584.0;tcp://10.65.2.3,10.65.34.3:59642;ud://2233744.284.1"
user1     339467 339457 99 Sep17 ?        18:58:25       /home/9999/software/interface-lammps-mlip-2/lmp_mpi -in in.nb_md
user1     339469 339457 99 Sep17 ?        18:59:50       /home/9999/software/interface-lammps-mlip-2/lmp_mpi -in in.nb_md
   :
user1     339531 339457 99 Sep17 ?        18:59:54       /home/9999/software/interface-lammps-mlip-2/lmp_mpi -in in.nb_md
```

Remote orted command spawns all local ranks for the MPI virtual machine

Putting it all together

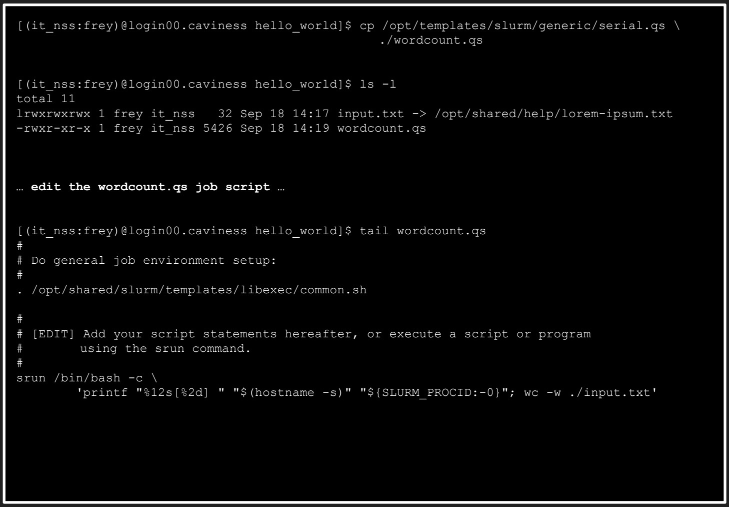
- Create a directory for the job
  - Plan ahead: organize work using a directory hierarchy
  - Input/data files
    - Use symbolic links when appropriate



```
[(it_nss:frey)@login00.caviness ~]$ mkdir -p ~/jobs/hello_world


[(it_nss:frey)@login00.caviness ~]$ cd ~/jobs/hello_world


[(it_nss:frey)@login00.caviness hello_world]$ ln -s /opt/shared/help/lorem-ipsum.txt \
                                                    ./input.txt
[(it_nss:frey)@login00.caviness hello_world]$ ls -l
total 1
lrwxrwxrwx 1 frey it_nss 32 Sep 18 14:17 input.txt -> /opt/shared/help/lorem-ipsum.txt
```

Putting it all together

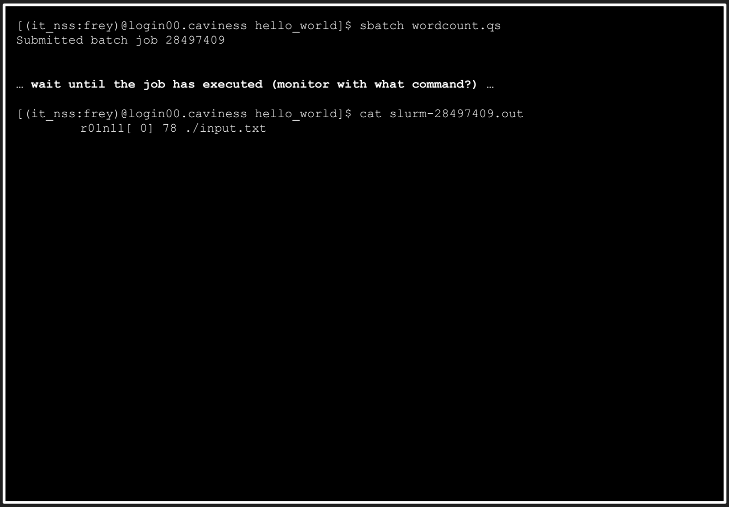
- Create a directory for the job
- Copy a job script template
  - Resource limits
  - Job properties (e.g. name)
  - Commands to be executed
  - Use srun to execute commands in *steps* with their own resource allocation/tracking



```
[(it_nss:frey)@login00.caviness hello_world]$ cp /opt/templates/slurm/generic/serial.qs \
                                                  ./wordcount.qs


[(it_nss:frey)@login00.caviness hello_world]$ ls -l
total 11
lrwxrwxrwx 1 frey it_nss   32 Sep 18 14:17 input.txt -> /opt/shared/help/lorem-ipsum.txt
-rwxr-xr-x 1 frey it_nss 5426 Sep 18 14:19 wordcount.qs



… edit the wordcount.qs job script …


[(it_nss:frey)@login00.caviness hello_world]$ tail wordcount.qs
#
# Do general job environment setup:
#
. /opt/shared/slurm/templates/libexec/common.sh

#
# [EDIT] Add your script statements hereafter, or execute a script or program
#        using the srun command.
#
srun /bin/bash -c \
        'printf "%12s[%2d] " "$(hostname -s)" "${SLURM_PROCID:-0}"; wc -w ./input.txt'
```

Putting it all together

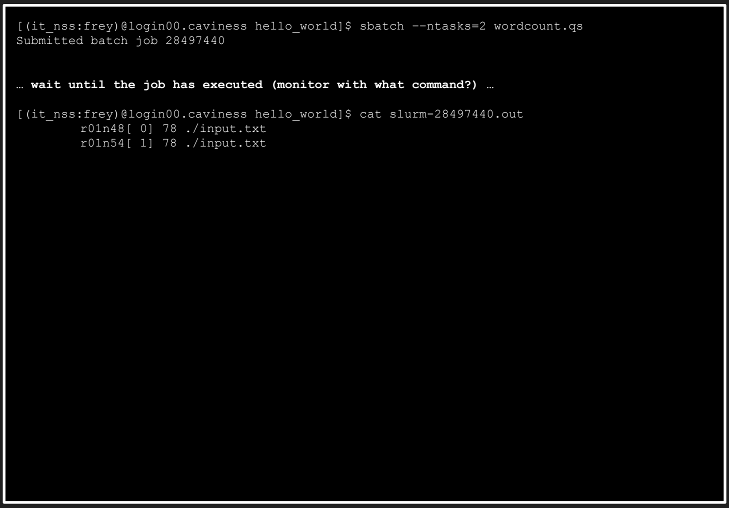
- Create a directory for the job
- Copy a job script template
- Submit the job



```
[(it_nss:frey)@login00.caviness hello_world]$ sbatch wordcount.qs
Submitted batch job 28497409

… wait until the job has executed (monitor with what command?) …

[(it_nss:frey)@login00.caviness hello_world]$ cat slurm-28497409.out
        r01n11[ 0] 78 ./input.txt
```

Putting it all together

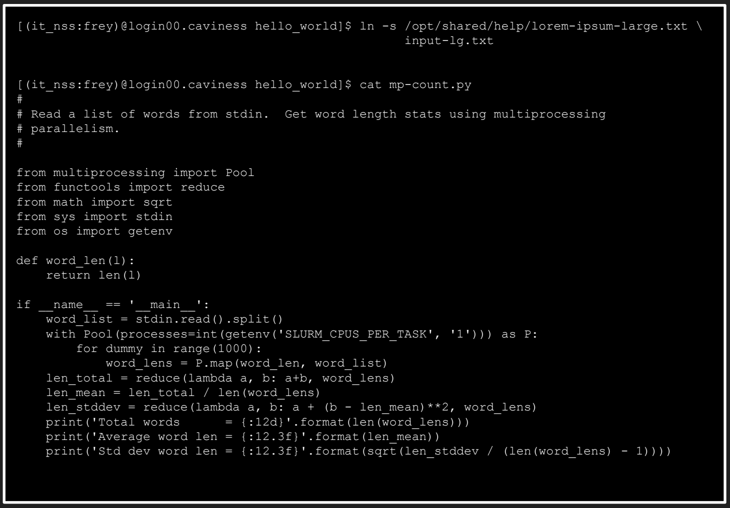
- Create a directory for the job
- Copy a job script template
- Submit the job
- Override resource limits
    - Limits in job script are overridden by options on the sbatch command itself



```
[(it_nss:frey)@login00.caviness hello_world]$ sbatch --ntasks=2 wordcount.qs
Submitted batch job 28497440


… wait until the job has executed (monitor with what command?) …

[(it_nss:frey)@login00.caviness hello_world]$ cat slurm-28497440.out
        r01n48[ 0] 78 ./input.txt
        r01n54[ 1] 78 ./input.txt
```

Putting it all together

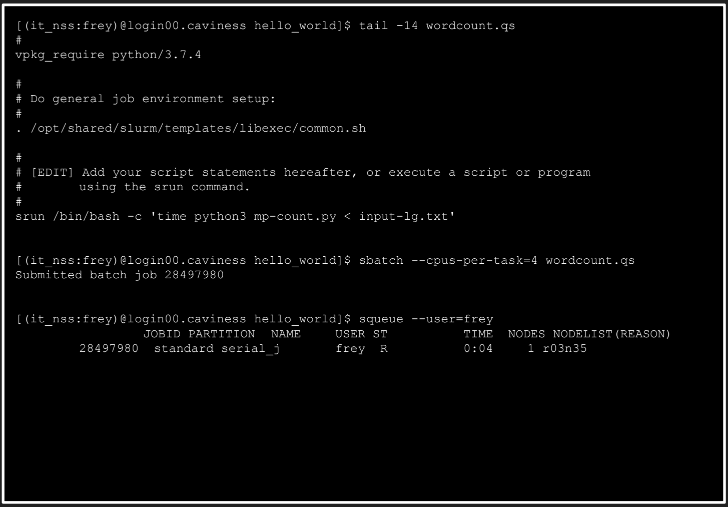
- Create a directory for the job
- Copy a job script template
- Submit the job
- Override resource limits
- Thread parallelism
    - Python multiprocessing with Slurm env vars controlling worker count



```
[(it_nss:frey)@login00.caviness hello_world]$ ln -s /opt/shared/help/lorem-ipsum-large.txt \
                                                    input-lg.txt


[(it_nss:frey)@login00.caviness hello_world]$ cat mp-count.py
#
# Read a list of words from stdin.  Get word length stats using multiprocessing
# parallelism.
#

from multiprocessing import Pool
from functools import reduce
from math import sqrt
from sys import stdin
from os import getenv

def word_len(l):
    return len(l)

if __name__ == '__main__':
    word_list = stdin.read().split()
    with Pool(processes=int(getenv('SLURM_CPUS_PER_TASK', '1'))) as P:
        for dummy in range(1000):
            word_lens = P.map(word_len, word_list)
    len_total = reduce(lambda a, b: a+b, word_lens)
    len_mean = len_total / len(word_lens)
    len_stddev = reduce(lambda a, b: a + (b - len_mean)**2, word_lens)
    print('Total words      = {:12d}'.format(len(word_lens)))
    print('Average word len = {:12.3f}'.format(len_mean))
    print('Std dev word len = {:12.3f}'.format(sqrt(len_stddev / (len(word_lens) - 1))))
```

Putting it all together

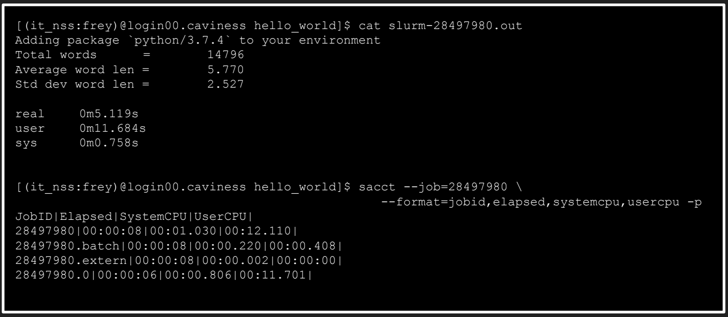
- Create a directory for the job
- Copy a job script template
- Submit the job
- Override resource limits
- Thread parallelism
  - Python multiprocessing with Slurm env vars controlling worker count
  - Technically NOT threaded…



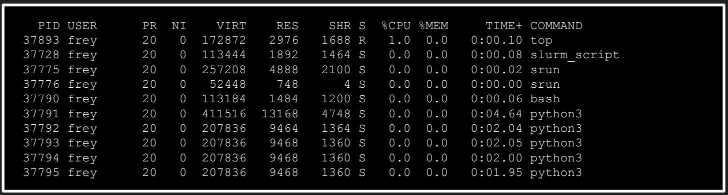
```
[(it_nss:frey)@login00.caviness hello_world]$ tail -14 wordcount.qs
#
vpkg_require python/3.7.4

#
# Do general job environment setup:
#
. /opt/shared/slurm/templates/libexec/common.sh

#
# [EDIT] Add your script statements hereafter, or execute a script or program
#        using the srun command.
#
srun /bin/bash -c 'time python3 mp-count.py < input-lg.txt'


[(it_nss:frey)@login00.caviness hello_world]$ sbatch --cpus-per-task=4 wordcount.qs
Submitted batch job 28497980


[(it_nss:frey)@login00.caviness hello_world]$ squeue --user=frey
                 JOBID PARTITION  NAME     USER ST          TIME  NODES NODELIST(REASON)
        28497980  standard serial_j       frey  R          0:04     1 r03n35
```

Putting it all together

- Create a directory for the job
- Copy a job script template
- Submit the job
- Override resource limits
- Thread parallelism
- Check resource usage



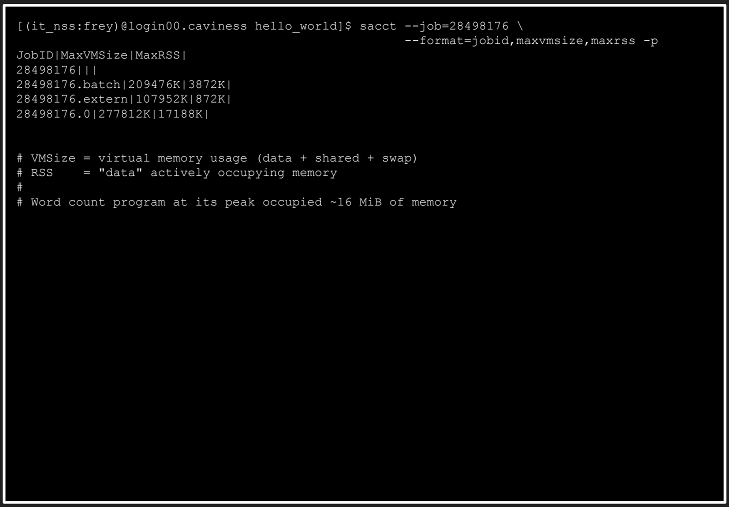
```
[(it_nss:frey)@login00.caviness hello_world]$ cat slurm-28497980.out
Adding package `python/3.7.4` to your environment
Total words      =        14796
Average word len =        5.770
Std dev word len =        2.527

real     0m5.119s
user     0m11.684s
sys      0m0.758s


[(it_nss:frey)@login00.caviness hello_world]$ sacct --job=28497980 \
                                                  --format=jobid,elapsed,systemcpu,usercpu -p
JobID|Elapsed|SystemCPU|UserCPU|
28497980|00:00:08|00:01.030|00:12.110|
28497980.batch|00:00:08|00:00.220|00:00.408|
28497980.extern|00:00:08|00:00.002|00:00:00|
28497980.0|00:00:06|00:00.806|00:11.701|
```



```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
37893 frey      20   0  172872   2976   1688 R   1.0  0.0   0:00.10 top
37728 frey      20   0  113444   1892   1464 S   0.0  0.0   0:00.08 slurm_script
37775 frey      20   0  257208   4888   2100 S   0.0  0.0   0:00.02 srun
37776 frey      20   0   52448    748      4 S   0.0  0.0   0:00.00 srun
37790 frey      20   0  113184   1484   1200 S   0.0  0.0   0:00.06 bash
37791 frey      20   0  411516  13168   4748 S   0.0  0.0   0:04.64 python3
37792 frey      20   0  207836   9464   1364 S   0.0  0.0   0:02.04 python3
37793 frey      20   0  207836   9468   1360 S   0.0  0.0   0:02.05 python3
37794 frey      20   0  207836   9468   1360 S   0.0  0.0   0:02.00 python3
37795 frey      20   0  207836   9468   1360 S   0.0  0.0   0:01.95 python3
```

Putting it all together

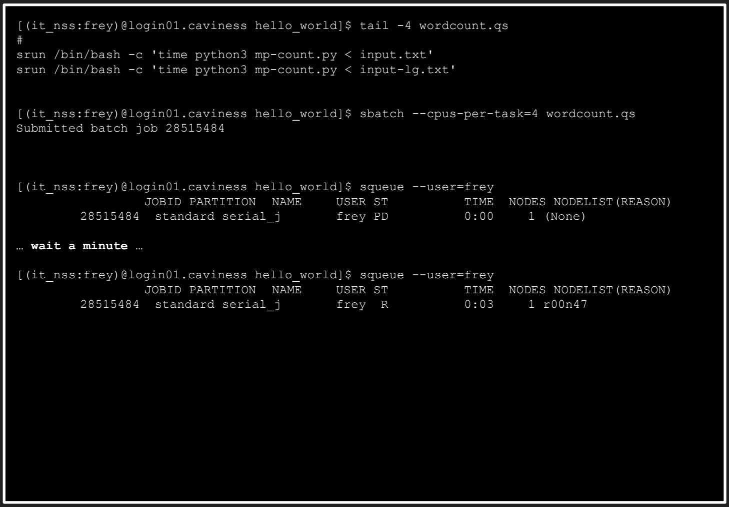
- Create a directory for the job
- Copy a job script template
- Submit the job
- Override resource limits
- Thread parallelism
- Check resource usage
  - Memory usage is captured periodically
  - Added a sleep() to Python



```
[(it_nss:frey)@login00.caviness hello_world]$ sacct --job=28498176 \
                                                     --format=jobid,maxvmsize,maxrss -p
JobID|MaxVMSize|MaxRSS|
28498176|||
28498176.batch|209476K|3872K|
28498176.extern|107952K|872K|
28498176.0|277812K|17188K|


# VMSize = virtual memory usage (data + shared + swap)
# RSS    = "data" actively occupying memory
#
# Word count program at its peak occupied ~16 MiB of memory
```

Putting it all together

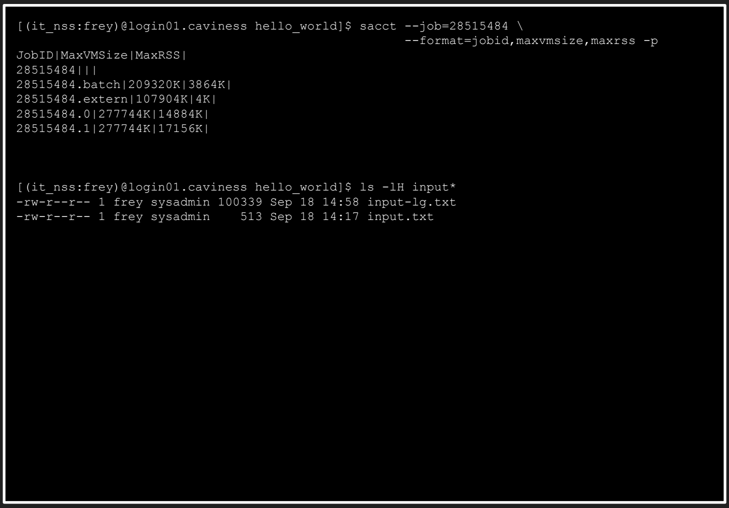
- Create a directory for the job
- Copy a job script template
- Submit the job
- Override resource limits
- Thread parallelism
- Check resource usage
- Repeat for two-step job



```
[(it_nss:frey)@login01.caviness hello_world]$ tail -4 wordcount.qs
#
srun /bin/bash -c 'time python3 mp-count.py < input.txt'
srun /bin/bash -c 'time python3 mp-count.py < input-lg.txt'


[(it_nss:frey)@login01.caviness hello_world]$ sbatch --cpus-per-task=4 wordcount.qs
Submitted batch job 28515484



[(it_nss:frey)@login01.caviness hello_world]$ squeue --user=frey
                 JOBID PARTITION   NAME     USER ST          TIME  NODES NODELIST(REASON)
          28515484  standard serial_j       frey PD          0:00     1  (None)

… wait a minute …

[(it_nss:frey)@login01.caviness hello_world]$ squeue --user=frey
                 JOBID PARTITION   NAME     USER ST          TIME  NODES NODELIST(REASON)
          28515484  standard serial_j       frey  R          0:03     1 r00n47
```

Putting it all together

- Create a directory for the job
- Copy a job script template
- Submit the job
- Override resource limits
- Thread parallelism
- Check resource usage
- Repeat for two-step job



```
[(it_nss:frey)@login01.caviness hello_world]$ sacct --job=28515484 \
                                                    --format=jobid,maxvmsize,maxrss -p
JobID|MaxVMSize|MaxRSS|
28515484|||
28515484.batch|209320K|3864K|
28515484.extern|107904K|4K|
28515484.0|277744K|14884K|
28515484.1|277744K|17156K|



[(it_nss:frey)@login01.caviness hello_world]$ ls -lH input*
-rw-r--r-- 1 frey sysadmin 100339 Sep 18 14:58 input-lg.txt
-rw-r--r-- 1 frey sysadmin    513 Sep 18 14:17 input.txt
```

Putting it all together

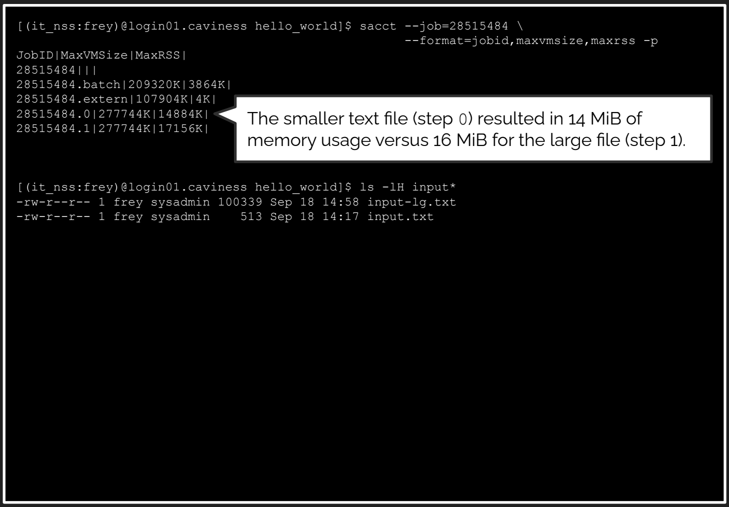
- Create a directory for the job
- Copy a job script template
- Submit the job
- Override resource limits
- Thread parallelism
- Check resource usage
- Repeat for two-step job

```
[(it_nss:frey)@login01.caviness hello_world]$ sacct --job=28515484 \
                                                     --format=jobid,maxvmsize,maxrss -p
JobID|MaxVMSize|MaxRSS|
28515484|||
28515484.batch|209320K|3864K|
28515484.extern|107904K|4K|
28515484.0|277744K|14884K|
28515484.1|277744K|17156K|


[(it_nss:frey)@login01.caviness hello_world]$ ls -lH input*
-rw-r--r-- 1 frey sysadmin 100339 Sep 18 14:58 input-lg.txt
-rw-r--r-- 1 frey sysadmin    513 Sep 18 14:17 input.txt
```

The `-H` *dereferences* symbolic links — shows info for the item to which they point, not the link itself

Putting it all together

- Create a directory for the job
- Copy a job script template
- Submit the job
- Override resource limits
- Thread parallelism
- Check resource usage
- Repeat for two-step job



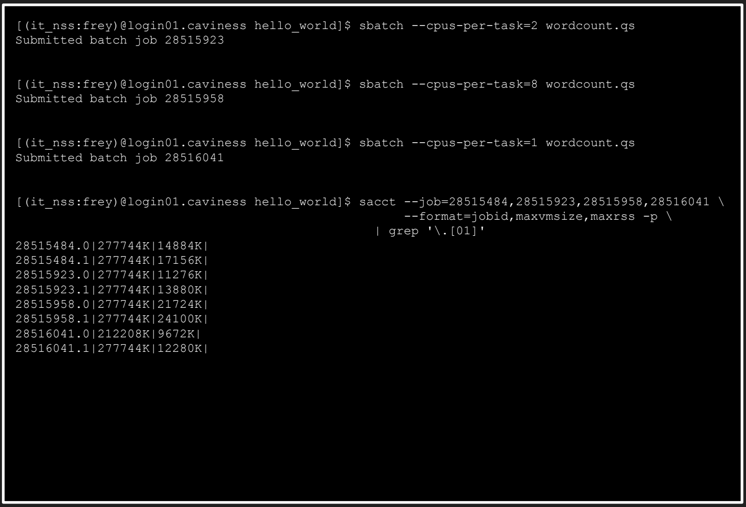
```
[(it_nss:frey)@login01.caviness hello_world]$ sacct --job=28515484 \
                                                    --format=jobid,maxvmsize,maxrss -p
JobID|MaxVMSize|MaxRSS|
28515484|||
28515484.batch|209320K|3864K|
28515484.extern|107904K|4K|
28515484.0|277744K|14884K|
28515484.1|277744K|17156K|


[(it_nss:frey)@login01.caviness hello_world]$ ls -lH input*
-rw-r--r-- 1 frey sysadmin 100339 Sep 18 14:58 input-lg.txt
-rw-r--r-- 1 frey sysadmin    513 Sep 18 14:17 input.txt
```

The smaller text file (step 0) resulted in 14 MiB of memory usage versus 16 MiB for the large file (step 1).

Files differ by 99826 bytes, or 97 KiB. Steps differ by 2272 KiB. Clearly there is a lot more overhead than just the text size.

Putting it all together

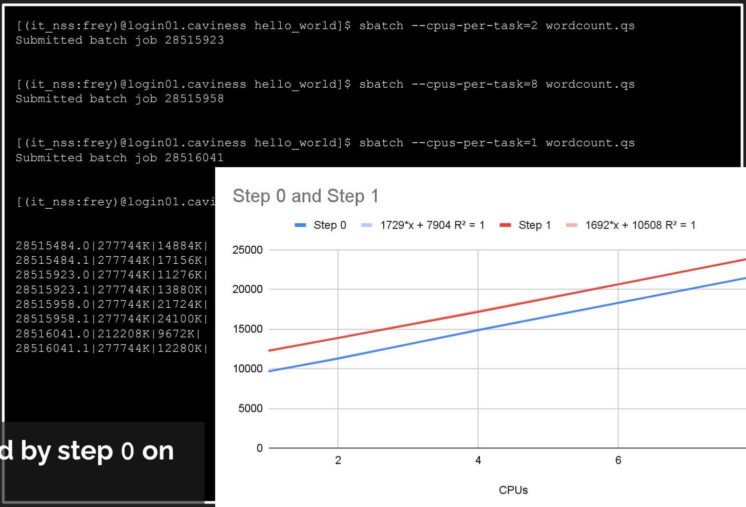
- Create a directory for the job
- Copy a job script template
- Submit the job
- Override resource limits
- Thread parallelism
- Check resource usage
- Repeat for two-step job



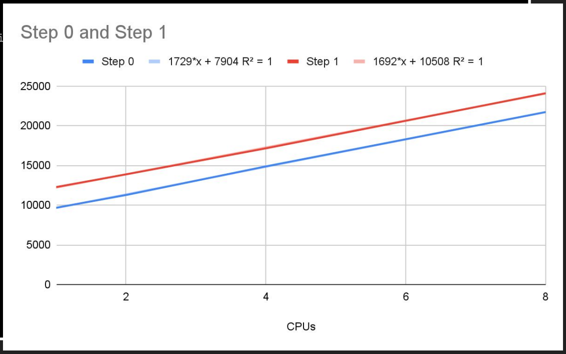
```
[(it_nss:frey)@login01.caviness hello_world]$ sbatch --cpus-per-task=2 wordcount.qs
Submitted batch job 28515923


[(it_nss:frey)@login01.caviness hello_world]$ sbatch --cpus-per-task=8 wordcount.qs
Submitted batch job 28515958


[(it_nss:frey)@login01.caviness hello_world]$ sbatch --cpus-per-task=1 wordcount.qs
Submitted batch job 28516041


[(it_nss:frey)@login01.caviness hello_world]$ sacct --job=28515484,28515923,28515958,28516041 \
                                                    --format=jobid,maxvmsize,maxrss -p \
                                                | grep '\.[01]'
28515484.0|277744K|14884K|
28515484.1|277744K|17156K|
28515923.0|277744K|11276K|
28515923.1|277744K|13880K|
28515958.0|277744K|21724K|
28515958.1|277744K|24100K|
28516041.0|212208K|9672K|
28516041.1|277744K|12280K|
```

Putting it all together

- Create a directory for the job
- Copy a job script template
- Submit the job
- Override resource limits
- Thread parallelism
- Check resource usage
- Repeat for two-step job



```
[(it_nss:frey)@login01.caviness hello_world]$ sbatch --cpus-per-task=2 wordcount.qs
Submitted batch job 28515923

[(it_nss:frey)@login01.caviness hello_world]$ sbatch --cpus-per-task=8 wordcount.qs
Submitted batch job 28515958

[(it_nss:frey)@login01.caviness hello_world]$ sbatch --cpus-per-task=1 wordcount.qs
Submitted batch job 28516041

[(it_nss:frey)@login01.cavi

28515484.0|277744K|14884K|
28515484.1|277744K|17156K|
28515923.0|277744K|11276K|
28515923.1|277744K|13880K|
28515958.0|277744K|21724K|
28515958.1|277744K|24100K|
28516041.0|212208K|9672K|
28516041.1|277744K|12280K|
```

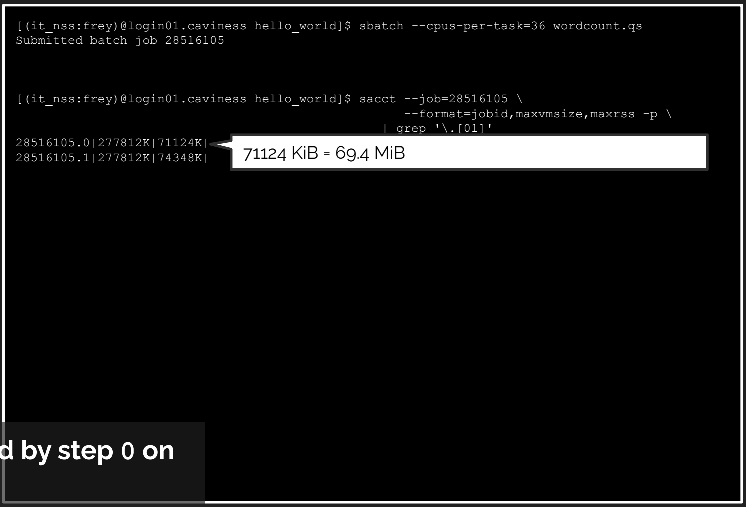


How much memory would be used by step 0 on all 36 CPUs?

Obvious linear relationship between multiprocessing worker count and memory usage.
1729 (36) + 7904 = 70148 = 68.5 MiB

Putting it all together

- Create a directory for the job
- Copy a job script template
- Submit the job
- Override resource limits
- Thread parallelism
- Check resource usage
- Repeat for two-step job



```
[(it_nss:frey)@login01.caviness hello_world]$ sbatch --cpus-per-task=36 wordcount.qs
Submitted batch job 28516105


[(it_nss:frey)@login01.caviness hello_world]$ sacct --job=28516105 \
                                                      --format=jobid,maxvmsize,maxrss -p \
                                                   | grep '\.[01]'
28516105.0|277812K|71124K|
28516105.1|277812K|74348K|
```

71124 KiB = 69.4 MiB

How much memory would be used by step 0 on all 36 CPUs?

UD IT Research Cyberinfrastructure

Obvious linear relationship between multiprocessing worker count and memory usage.
1729 (36) + 7904 = 70148 = 68.5 MiB

Seeking help?

- Documentation available in several forms
  - `--help` flag to the Slurm commands
  - `man <command>` to display Slurm manual pages for commands
  - IT RCI wiki
    - https://docs.hpc.udel.edu/abstract/caviness/runjobs/runjobs
    - https://docs.hpc.udel.edu/abstract/darwin/runjobs/runjobs
- Speak to coworkers who have experience on DARWIN
  - Try to shepherd your group's computational knowledge from one generation of members to the next
- Submit questions to the HPC community, **`hpc-ask@udel.edu`**
  - Usage of particular software, discussion of algorithms, etc.
  - Anything not related to the cluster hardware

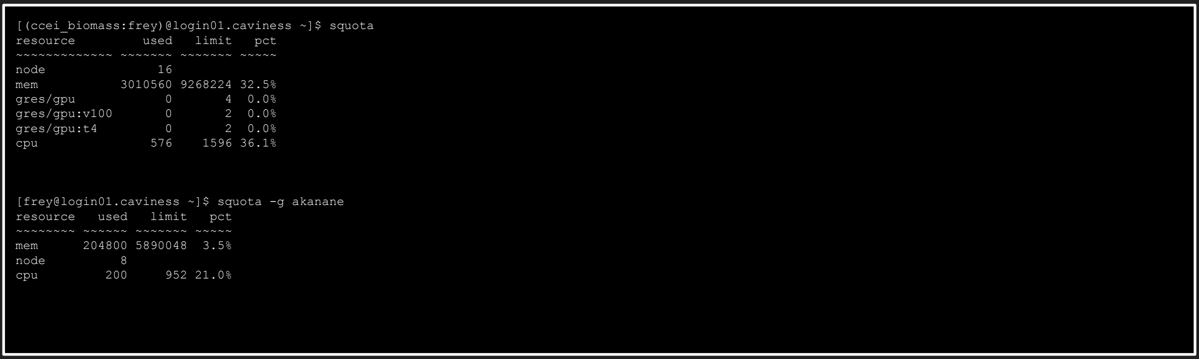
Additional Topics

Caviness: Workgroup resource quotas

- Faculty stakeholders buy-in to Caviness
  - Subsidized purchase of node, storage resources
  - Each stakeholder's total expenditure ⇒ fraction of all resources
    - Resource fraction ⇒ Slurm workgroup share ⇒ scheduling priority weight
- Each stakeholder gets a workgroup partition
  - Backed by all nodes of the type(s) purchased
  - A *resource quota* limits total CPU/GPU count, memory actively used by jobs
  - Use partition name "_WORKGROUP_" when submitting jobs

Caviness: Workgroup resource quotas

- Check current resource quota usage
  - Defaults to shell's current workgroup
  - Arbitrary workgroup can be indicated with "-g" option



```
[(ccei_biomass:frey)@login01.caviness ~]$ squota
resource         used   limit   pct
~~~~~~~~~~~~~ ~~~~~~~ ~~~~~~~ ~~~~~
node               16
mem           3010560 9268224 32.5%
gres/gpu            0       4  0.0%
gres/gpu:v100       0       2  0.0%
gres/gpu:t4         0       2  0.0%
cpu               576    1596 36.1%



[frey@login01.caviness ~]$ squota -g akanane
resource   used   limit   pct
~~~~~~~~ ~~~~~~ ~~~~~~~ ~~~~~
mem      204800 5890048  3.5%
node          8
cpu         200     952 21.0%
```

DARWIN: Working with allocations

- Workgroups receive a time- and capacity-limited share of resources
    - A *service unit* (SU) equates with e.g. CPU•hour, GPU•hour
- Jobs submitted using workgroup *W* are billed against that allocation
- As jobs are executed:
    - Prolog: the total SU required to execute the job is pre-debited from allocation
        - E.g. (job time limit) • (CPU count)
        - Insufficient funds, job is denied
    - Epilog: the SU usage is adjusted and debited
        - E.g. (job wall time) • (CPU count)
        - Failure due to hardware issues = not debited

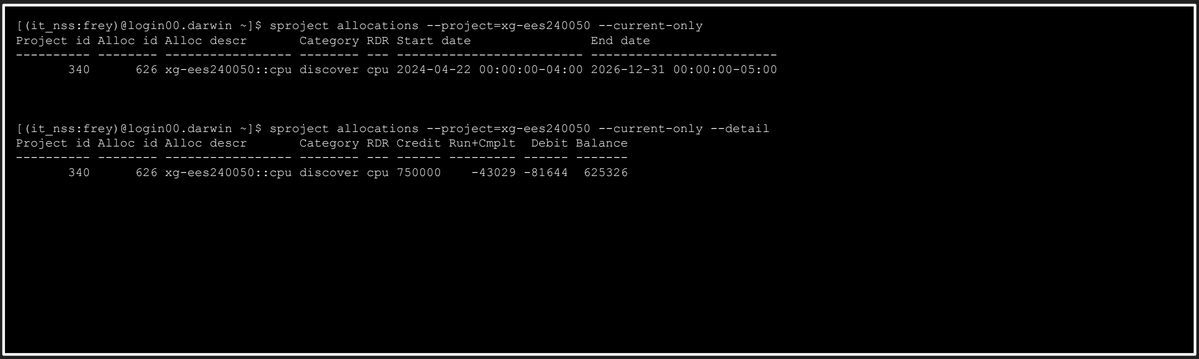
DARWIN: Working with allocations

- No workgroup partitions as on Caviness
  - Partitions target types of nodes in DARWIN
  - User must match resource needs of job to appropriate partition

# nodes	Partition	CPU / node	Memory / node	HW / node
48	standard	64	512 GiB	
32	large-mem	64	1024 GiB	
11	xlarge-mem	64	2048 GiB	
1	extended-mem	64	1024 GiB	2.8 TB swap
9	gpu-t4	64	512 GiB	1 x T4
3	gpu-v100	48	768 GiB	3 x V100
1	gpu-mi50	64	512 GiB	1 x Mi50
1	gpu-mi100	128	512 GiB	1 x Mi100

DARWIN: Working with allocations

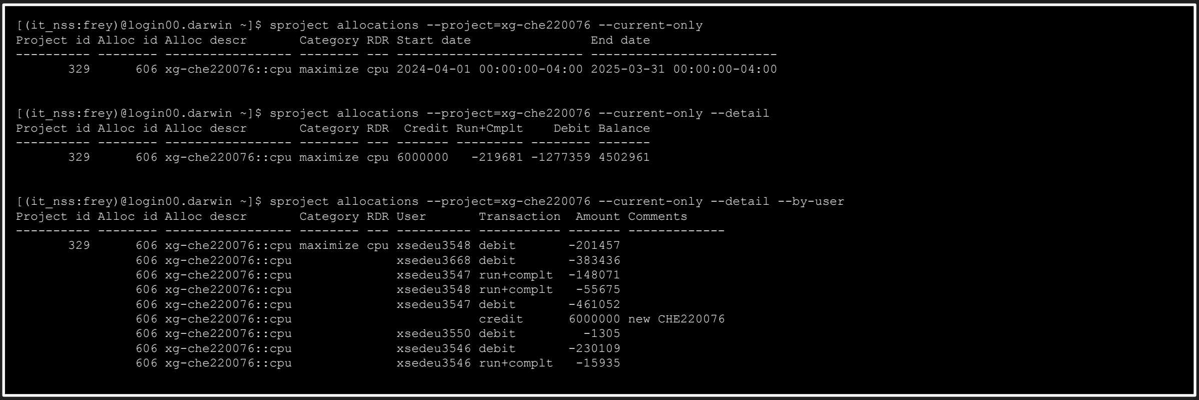
- Check allocation status
  - Without "--current-only" full history of workgroup allocations is displayed
  - The "--detail" flag shows SU balance for each allocation



```
[(it_nss:frey)@login00.darwin ~]$ sproject allocations --project=xg-ees240050 --current-only
Project id Alloc id Alloc descr       Category RDR Start date                End date
---------- -------- ----------------- -------- --- ------------------------- -------------------------
       340      626 xg-ees240050::cpu discover cpu 2024-04-22 00:00:00-04:00 2026-12-31 00:00:00-05:00


[(it_nss:frey)@login00.darwin ~]$ sproject allocations --project=xg-ees240050 --current-only --detail
Project id Alloc id Alloc descr       Category RDR Credit Run+Cmplt  Debit Balance
---------- -------- ----------------- -------- --- ------ --------- ------ -------
       340      626 xg-ees240050::cpu discover cpu 750000    -43029 -81644  625326
```

DARWIN: Working with allocations

- Check allocation status
  - The "--by-user" flag shows credits and debits resolved to each group member



```
[(it_nss:frey)@login00.darwin ~]$ sproject allocations --project=xg-che220076 --current-only
Project id Alloc id Alloc descr       Category RDR Start date                End date
---------- -------- ----------------- -------- --- ------------------------- -------------------------
       329      606 xg-che220076::cpu maximize cpu 2024-04-01 00:00:00-04:00 2025-03-31 00:00:00-04:00


[(it_nss:frey)@login00.darwin ~]$ sproject allocations --project=xg-che220076 --current-only --detail
Project id Alloc id Alloc descr       Category RDR  Credit Run+Cmplt    Debit Balance
---------- -------- ----------------- -------- --- ------- --------- -------- -------
       329      606 xg-che220076::cpu maximize cpu 6000000   -219681 -1277359 4502961


[(it_nss:frey)@login00.darwin ~]$ sproject allocations --project=xg-che220076 --current-only --detail --by-user
Project id Alloc id Alloc descr       Category RDR User       Transaction  Amount Comments
---------- -------- ----------------- -------- --- ---------- ----------- ------- -------------
       329      606 xg-che220076::cpu maximize cpu xsedeu3548 debit       -201457
                606 xg-che220076::cpu              xsedeu3668 debit       -383436
                606 xg-che220076::cpu              xsedeu3547 run+complt  -148071
                606 xg-che220076::cpu              xsedeu3548 run+complt   -55675
                606 xg-che220076::cpu              xsedeu3547 debit       -461052
                606 xg-che220076::cpu                         credit      6000000 new CHE220076
                606 xg-che220076::cpu              xsedeu3550 debit         -1305
                606 xg-che220076::cpu              xsedeu3546 debit       -230109
                606 xg-che220076::cpu              xsedeu3546 run+complt   -15935
```