

Getting Started with Batch Job Scheduling: Slurm Edition

COMPLECS Team

<https://bit.ly/COMPLECS>

<https://github.com/sdsc-complecs>

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

About COMPLECS

COMPLECS (COMPrehensive Learning for end-users to Effectively utilize CyberinfraStructure) is a new SDSC program where training will cover ***non-programming*** skills needed to effectively use supercomputers. Topics include parallel computing concepts, Linux tools and bash scripting, security, batch and interactive computing, how to get help, and data management.

COMPLECS is supported by NSF [CISE/OAC-2320934](#)



COMPLECS: Batch Computing Series

Working with the Linux Scheduler

How to interact with the Linux scheduler and run your research workloads on your personal computer, a shared workstation, or even a high-performance computing system.

Getting Started with Batch Job Scheduling: Slurm Edition

How to schedule your batch jobs on high-performance computing systems using the Slurm Workload Manager.

High-Throughput and Many-Task Computing Workflows: Slurm Edition

How to build and run your high-throughput and many-task computing workflows on high-performance computing systems using the Slurm Workload Manager.





Getting Started with Batch Job Scheduling: Slurm Edition

Learning Goals:

- What is a batch job?
- What is Slurm?
- How to start interacting with Slurm
- How to write, submit and run your first batch job on a high-performance computing system

Prerequisite Knowledge:

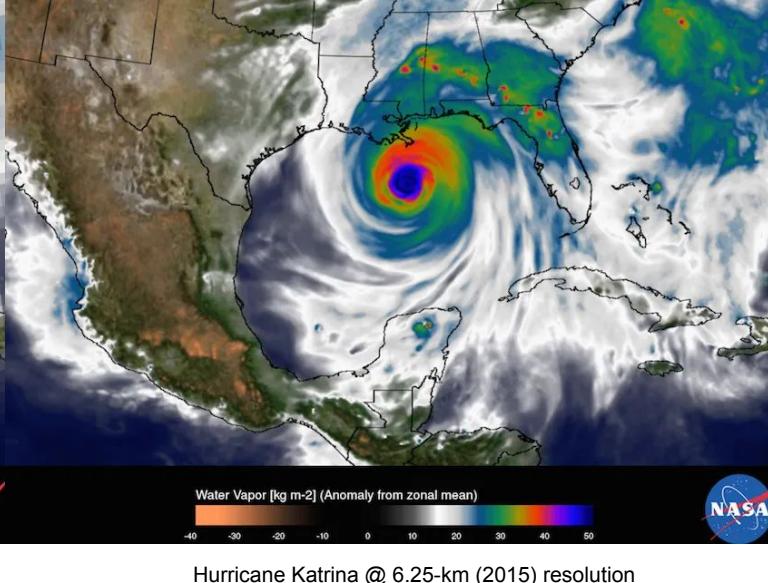
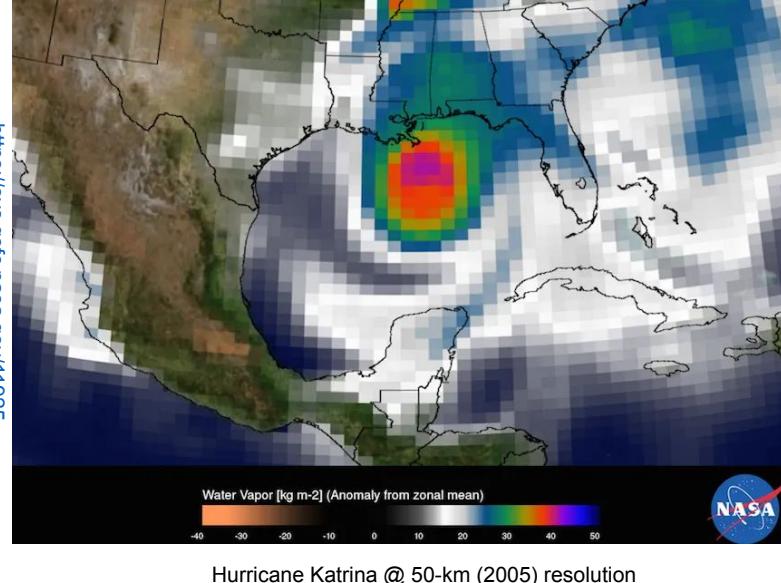
- [Parallel Computing Concepts](#)
- [Intermediate Linux and Shell Scripting](#)

Table of Contents

- **High-Performance Computing and System Architecture**
- **Batch Job Scheduling and Resource Management**
- **Getting Started with the Slurm Workload Manager**
 - `squeue` - view information about jobs located in the Slurm scheduling queue
 - `sinfo` - view information about Slurm nodes and partitions
- **First Batch Job(s) with Slurm**
 - `sbatch` - submit a batch script to Slurm
 - `scancel` - used to signal or cancel jobs, job arrays or job steps
 - `sacct` - displays accounting data for all jobs in the Slurm database
- **Best Practices with Slurm**
- **Questions & Answers (30 min)**

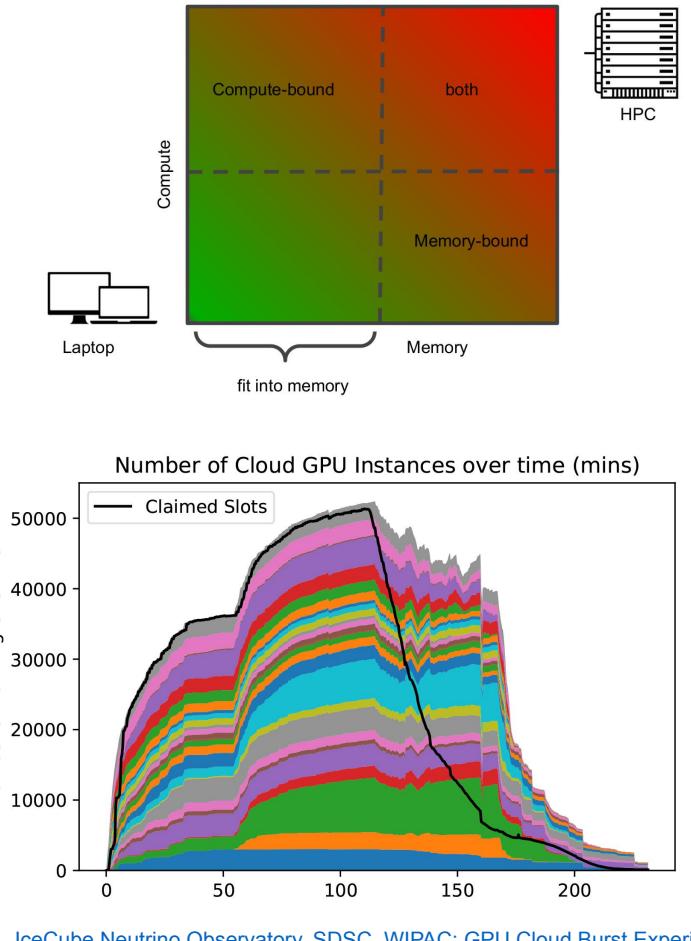
High-Performance Computing (HPC)

High-performance computing (HPC) uses supercomputers and computer clusters to solve advanced computation problems.

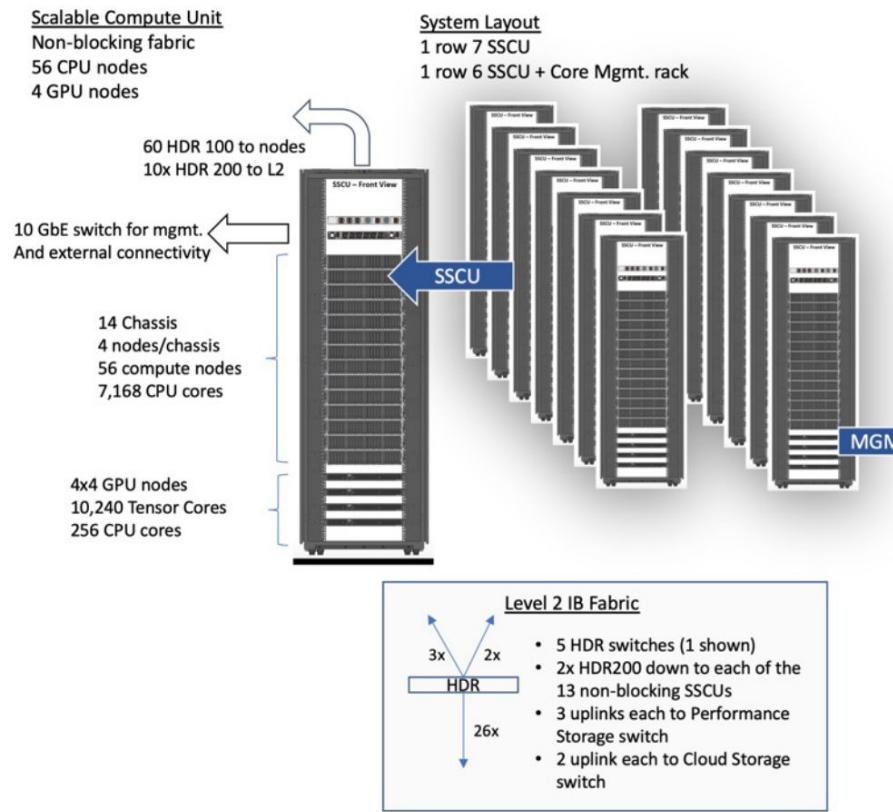


Key advantages of leveraging HPC for your research:

- **Speed** - Solve a problem more quickly
- **Scale** - Solve a larger, more complex problem
- **Throughput** - Solve many (simple) problems more quickly

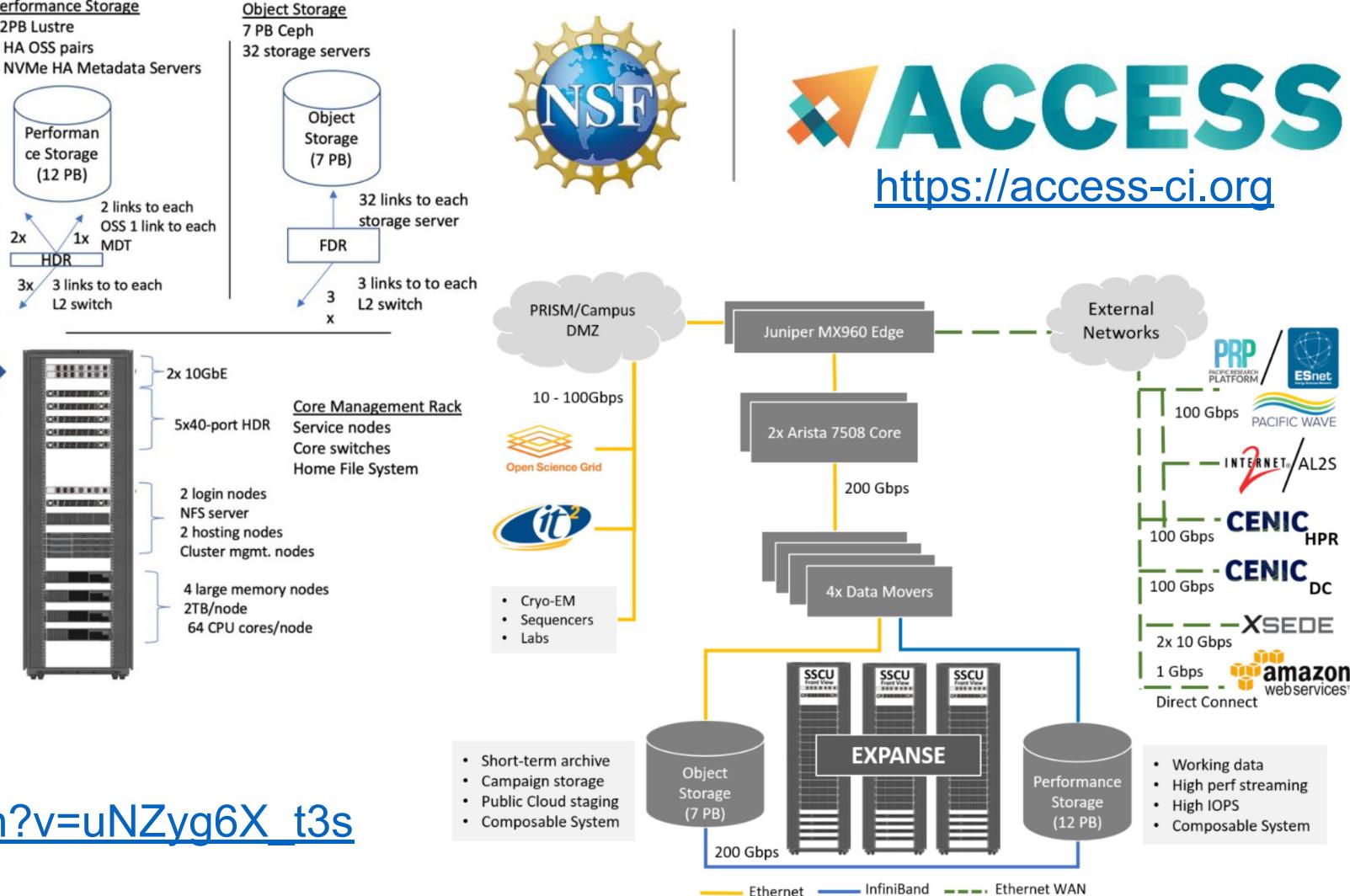


HPC System Architecture: Expanse @ SDSC



<https://expanse.sdsc.edu>

https://www.youtube.com/watch?v=uNZyg6X_t3s



HPC Compute Node: Dell C4140



Figure 1. Front view of the system

1. Control panel

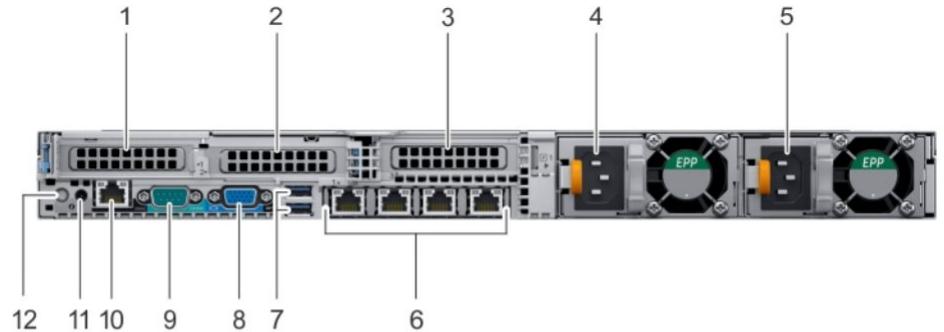
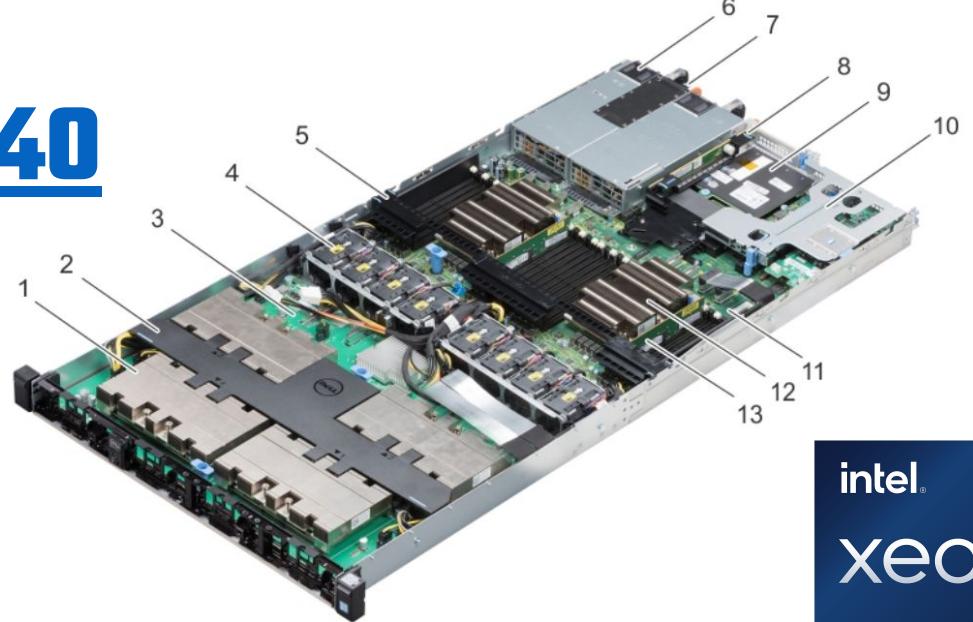


Figure 2. Rear view of the system

1. PCIe expansion card slot 1
 2. PCIe expansion card slot 2
 3. PCIe expansion card slot 3
 4. Power supply unit (PSU 1)
 5. Power supply unit (PSU 2)
 6. Ethernet connectors (4)
 7. USB connectors (2)
 9. Serial connector
 10. iDRAC Enterprise port
 11. NMI button
 12. System identification button
- NOTE:** This slot is dedicated for BOSS configuration.



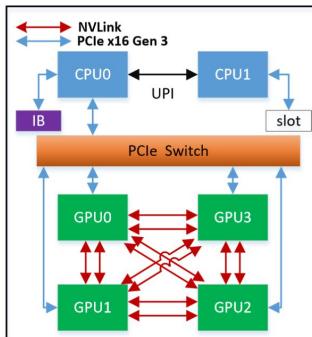
[Intel Xeon Gold 6248](#)

Figure 4. Configuration K

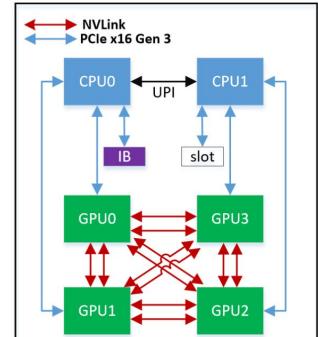
1. NVLink Heat sink and processor (4)
2. NVLink air shroud
3. NVLink board
4. Cooling fan (8)
5. Air shroud
6. PSU (2)
7. Information tag
9. Network daughter card (NDC)
11. System board
12. Processor and heat sink (2)
13. DIMMs (24)



[NVIDIA V100 32GB SMX2](#)



(a) Configuration K



(b) Configuration M

Figure 1: The comparison between configuration K and configuration M in C4140 server

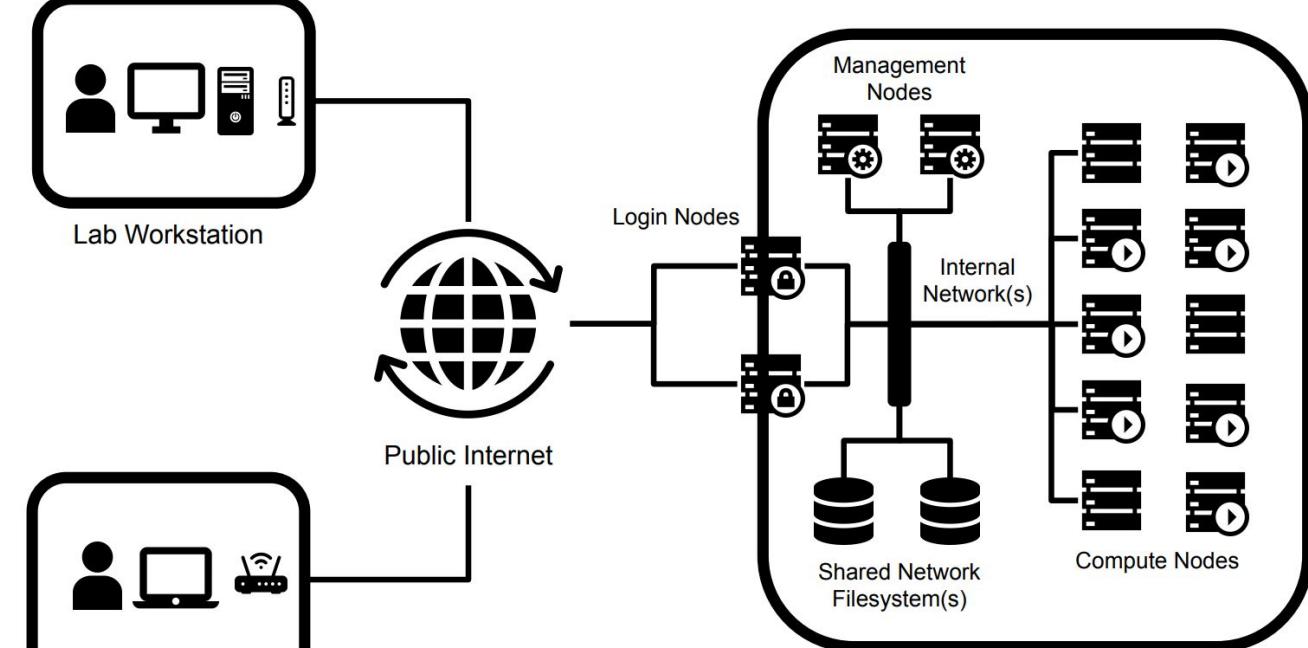
HPC System Architecture: Conceptual Model

Login node(s): Provide remote access to an HPC system; use only for simple tasks such as editing files, limited data transfers to and from the system, and batch job submission

Compute nodes: Run computational workloads: simulations, data analysis and visualization

Internal Network(s): Provide high-bandwidth, low-latency communication between compute nodes ; access to shared (parallel) filesystems; system management

Shared Network Filesystem(s): Provide input/output (I/O) access to data storage systems from any compute node



Management node(s): Run core system services such as cluster management software, system monitoring software, *batch job scheduler*, etc

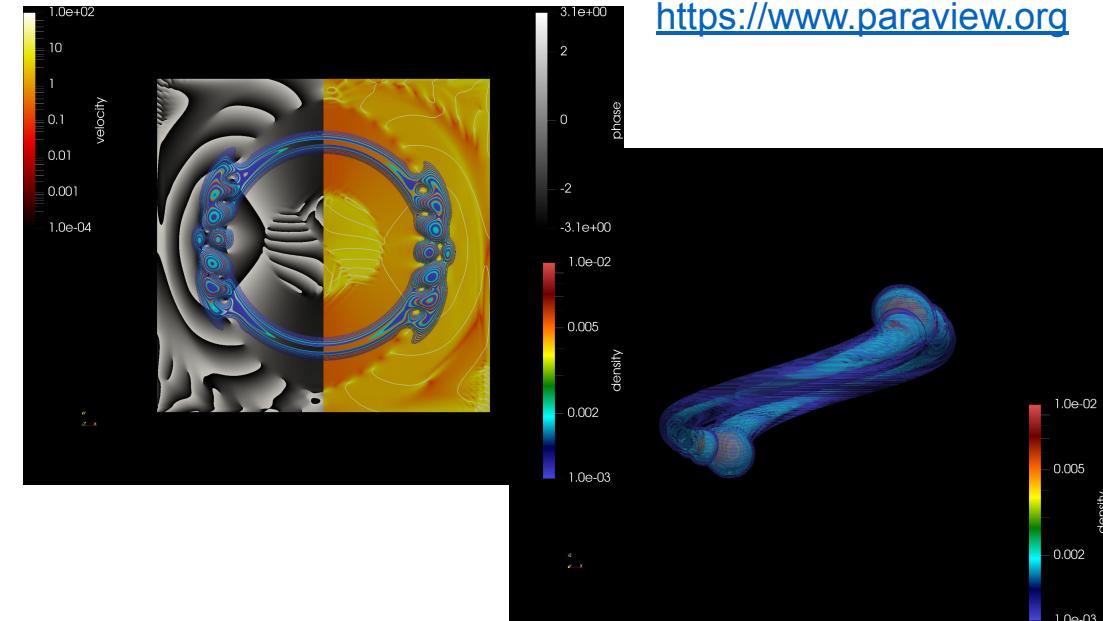
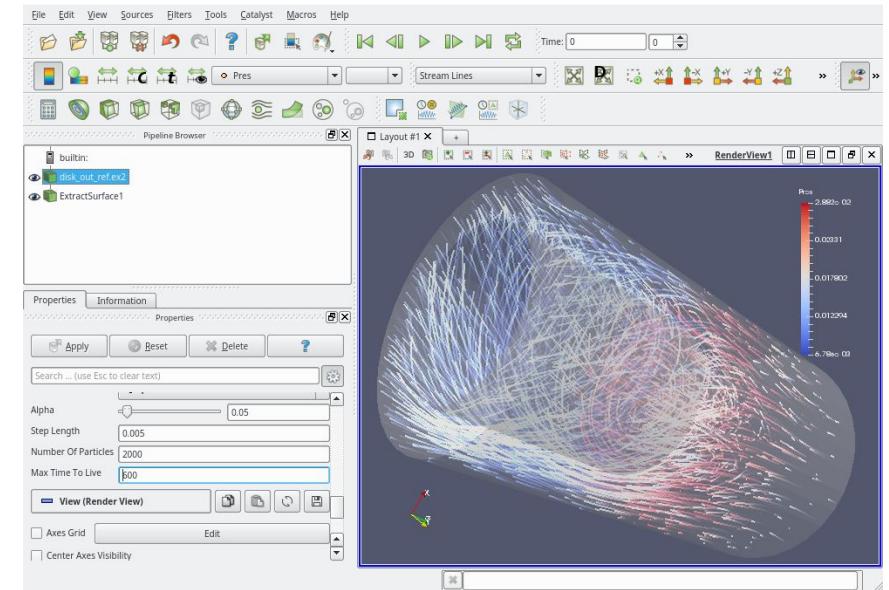
Table of Contents

- High-Performance Computing and System Architecture
- Batch Job Scheduling and Resource Management
- Getting Started with the Slurm Workload Manager
 - `squeue` - view information about jobs located in the Slurm scheduling queue
 - `sinfo` - view information about Slurm nodes and partitions
- First Batch Job(s) with Slurm
 - `sbatch` - submit a batch script to Slurm
 - `scancel` - used to signal or cancel jobs, job arrays or job steps
 - `sacct` - displays accounting data for all jobs in the Slurm database
- Best Practices with Slurm
- Questions & Answers (30 min)

Interactive vs. Batch Computing

Interactive computing is the use of computer programs that accept input from the user as they execute. Examples in HPC include the use of software applications for code development and performance optimization as well as real-time data exploration, analysis and visualization.

Batch processing is the execution of a series of computer programs, known as jobs, on a system without user interaction. Batch computing systems are widely used to automate high-volume, repetitive tasks such as data processing and to allow the shared use of advanced compute resources like an HPC system among many users.



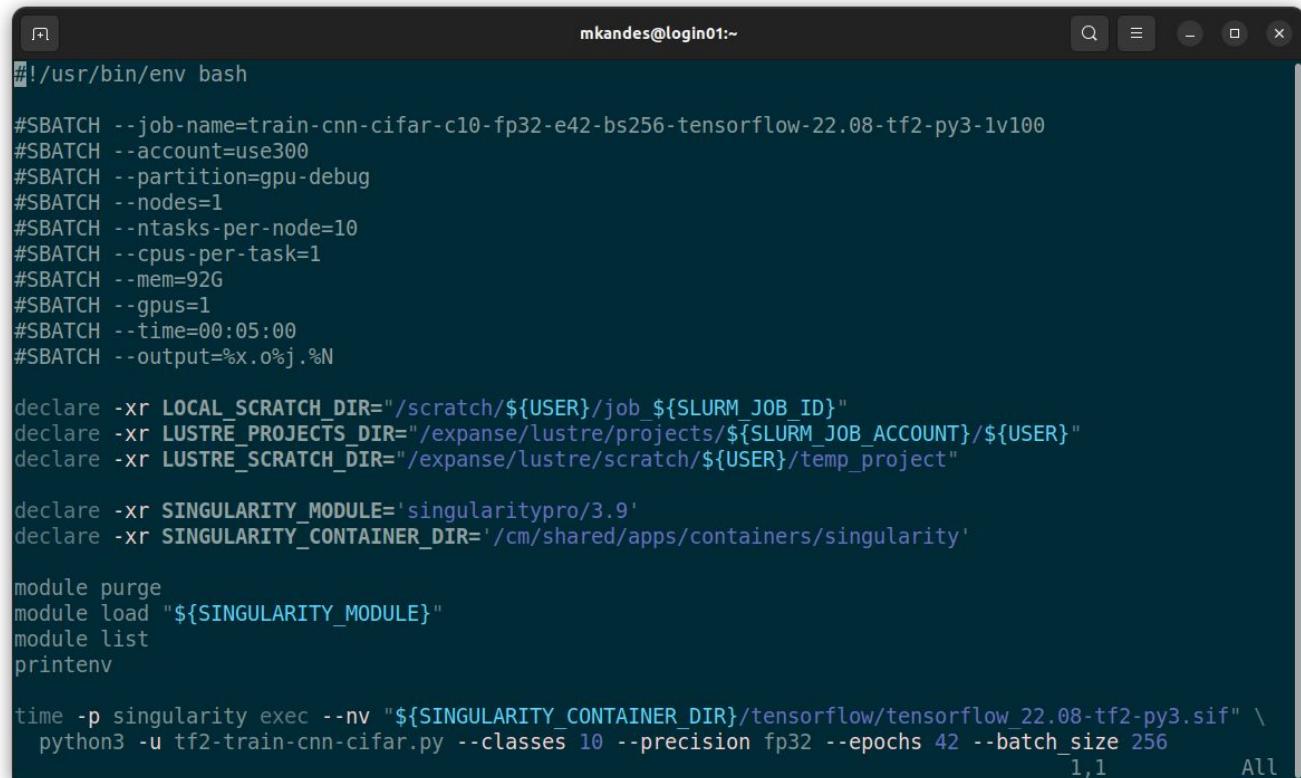
<https://www.paraview.org>

What is a Batch Job?

A **batch job** is a pre-scripted sets of commands that are executed as processes, which may or may not be multi-threaded, on a certain type or set of dedicated compute resources within a computing system for a given amount of time without user interaction.

On HPC systems, batch jobs may run large-scale simulations, perform complex data analysis, or any other types of demanding computational tasks.

A batch job is specified as a standard shell script that is then submitted to a queue of other batch jobs managed by a scheduler, which assigns the job to available resources and ensures that the job runs to completion.



The screenshot shows a terminal window titled 'mkandes@login01:~'. The window contains a bash script with the following content:

```
#!/usr/bin/env bash

#SBATCH --job-name=train-cnn-cifar-c10-fp32-e42-bs256-tensorflow-22.08-tf2-py3-1v100
#SBATCH --account=use300
#SBATCH --partition=gpu-debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=10
#SBATCH --cpus-per-task=1
#SBATCH --mem=92G
#SBATCH --gpus=1
#SBATCH --time=00:05:00
#SBATCH --output=%x.o%j.%N

declare -xr LOCAL_SCRATCH_DIR="/scratch/${USER}/job_${SLURM_JOB_ID}"
declare -xr LUSTRE_PROJECTS_DIR="/expanse/lustre/projects/${SLURM_JOB_ACCOUNT}/${USER}"
declare -xr LUSTRE_SCRATCH_DIR="/expanse/lustre/scratch/${USER}/temp_project"

declare -xr SINGULARITY_MODULE='singularitypro/3.9'
declare -xr SINGULARITY_CONTAINER_DIR='/cm/shared/apps/containers/singularity'

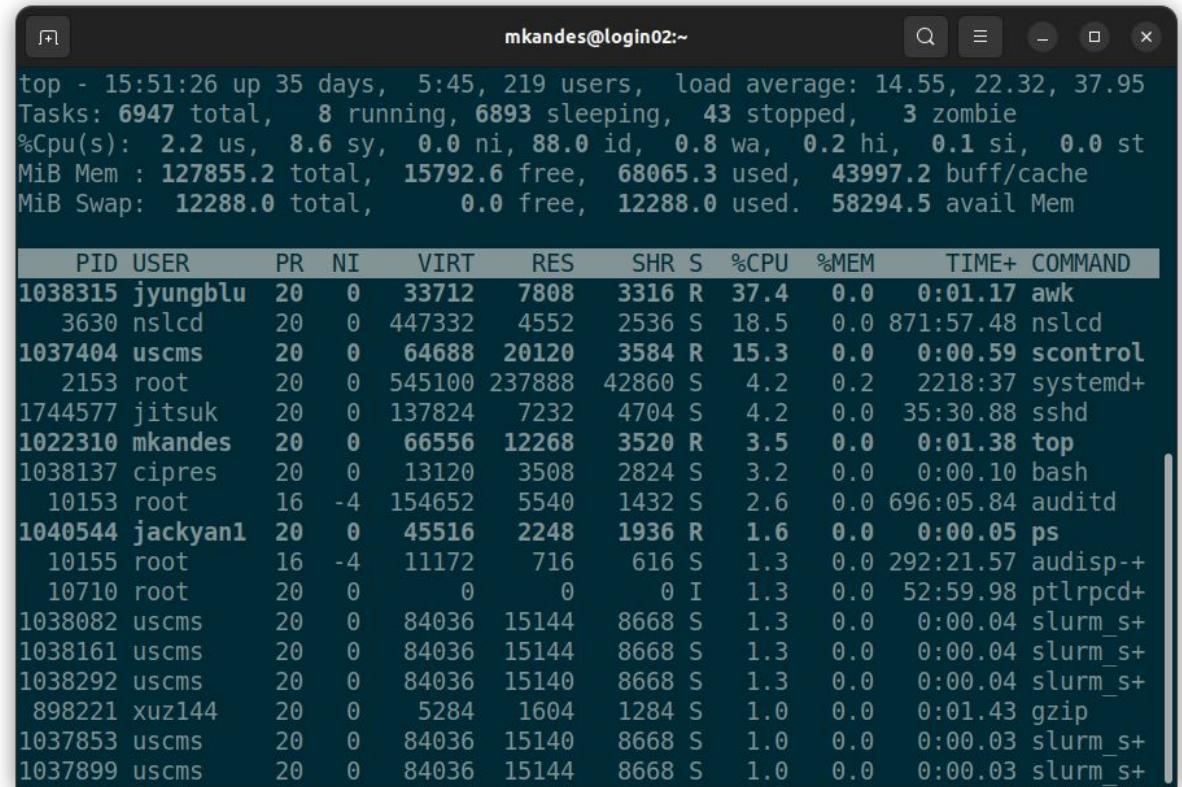
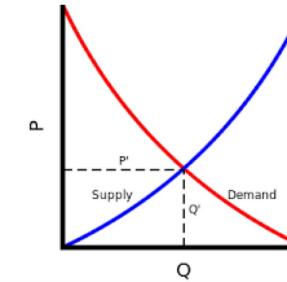
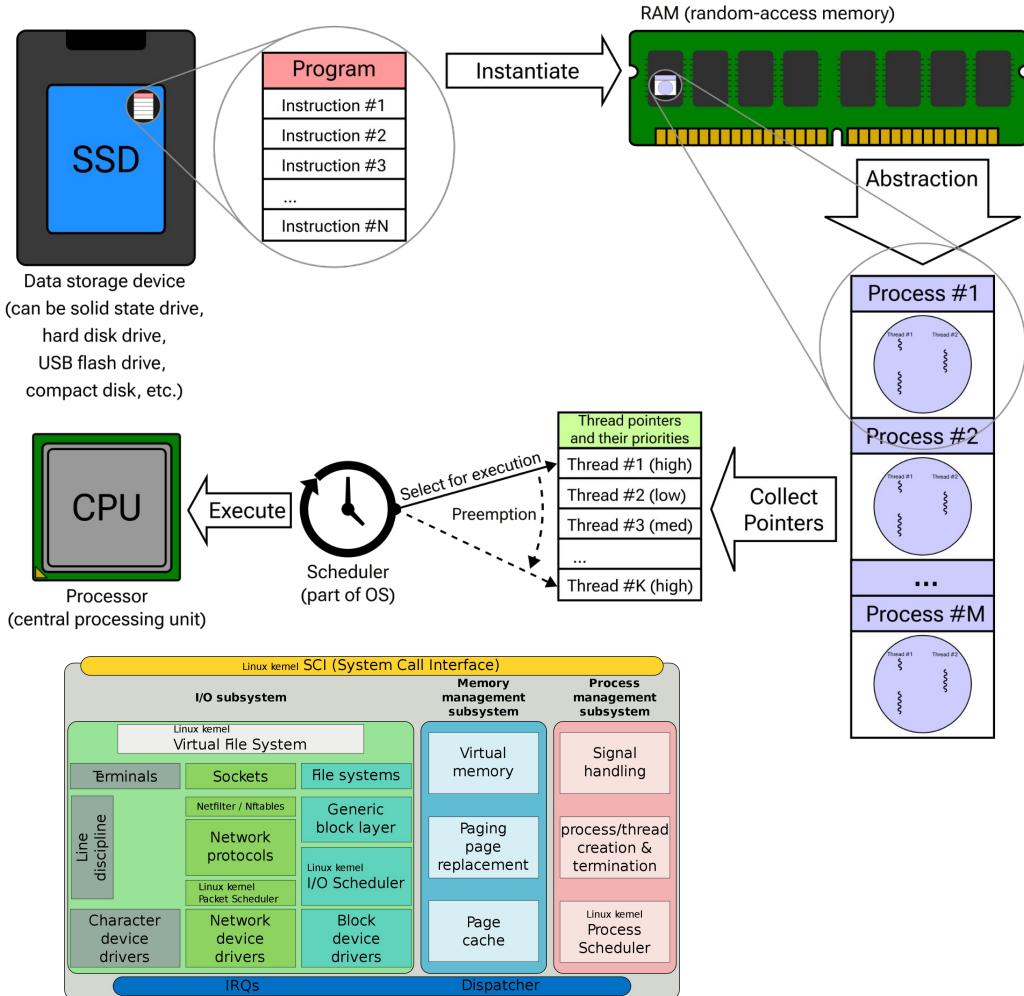
module purge
module load "${SINGULARITY_MODULE}"
module list
printenv

time -p singularity exec --nv "${SINGULARITY_CONTAINER_DIR}/tensorflow/tensorflow_22.08-tf2-py3.sif" \
    python3 -u tf2-train-cnn-cifar.py --classes 10 --precision fp32 --epochs 42 --batch_size 256
```

The command 'time' is used to measure the execution time of the singularity command. The output of the command is shown at the bottom right of the terminal window.

Scheduling

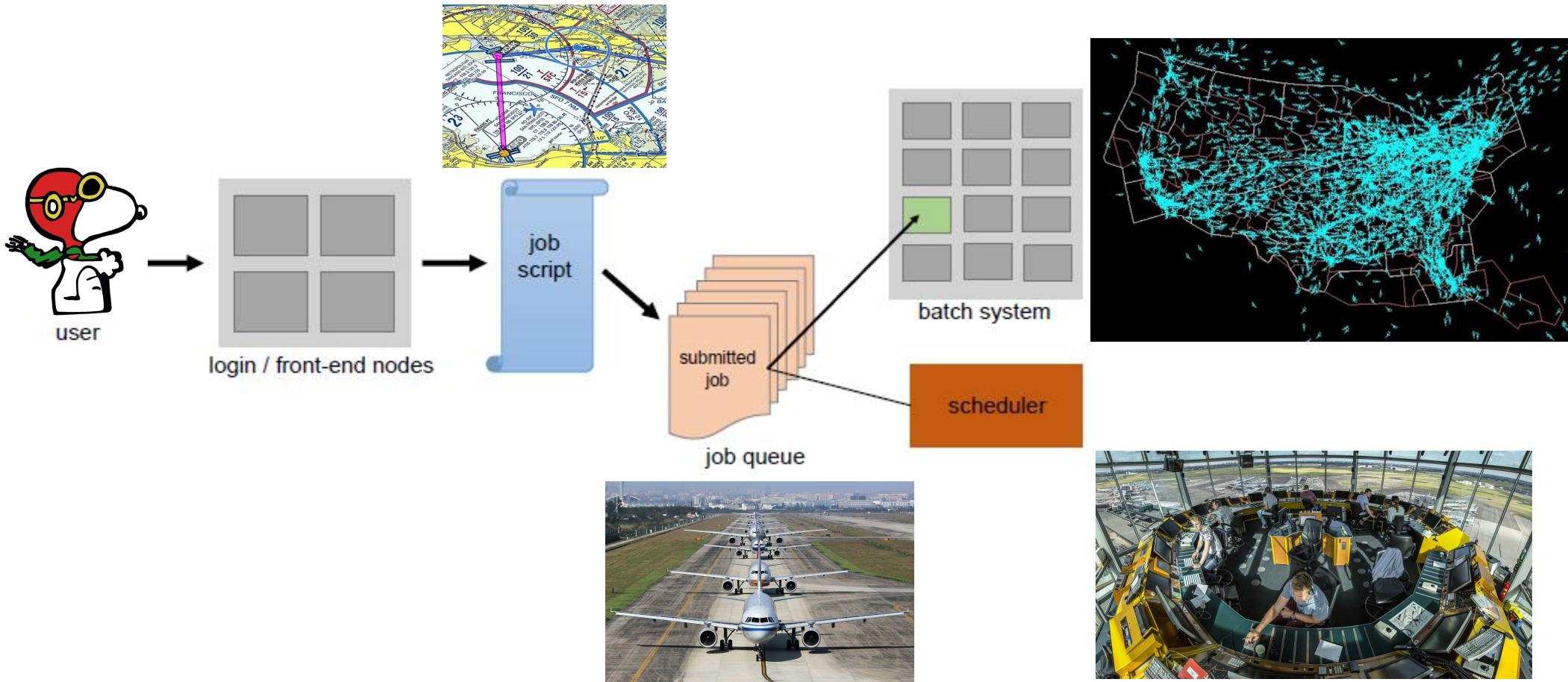
Scheduling is the action of assigning resources to perform tasks.



[top](#) and [htop](#) are useful interactive tools for inspecting and monitoring running processes on Unix-like operating systems.

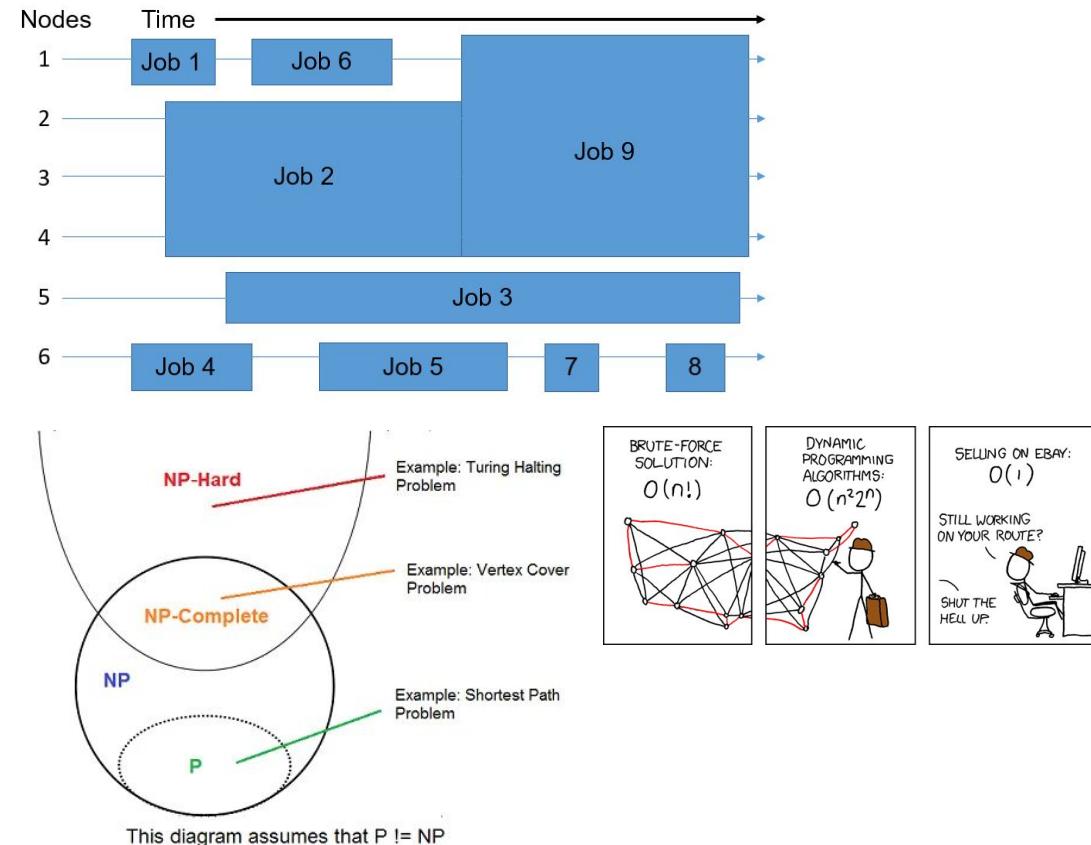
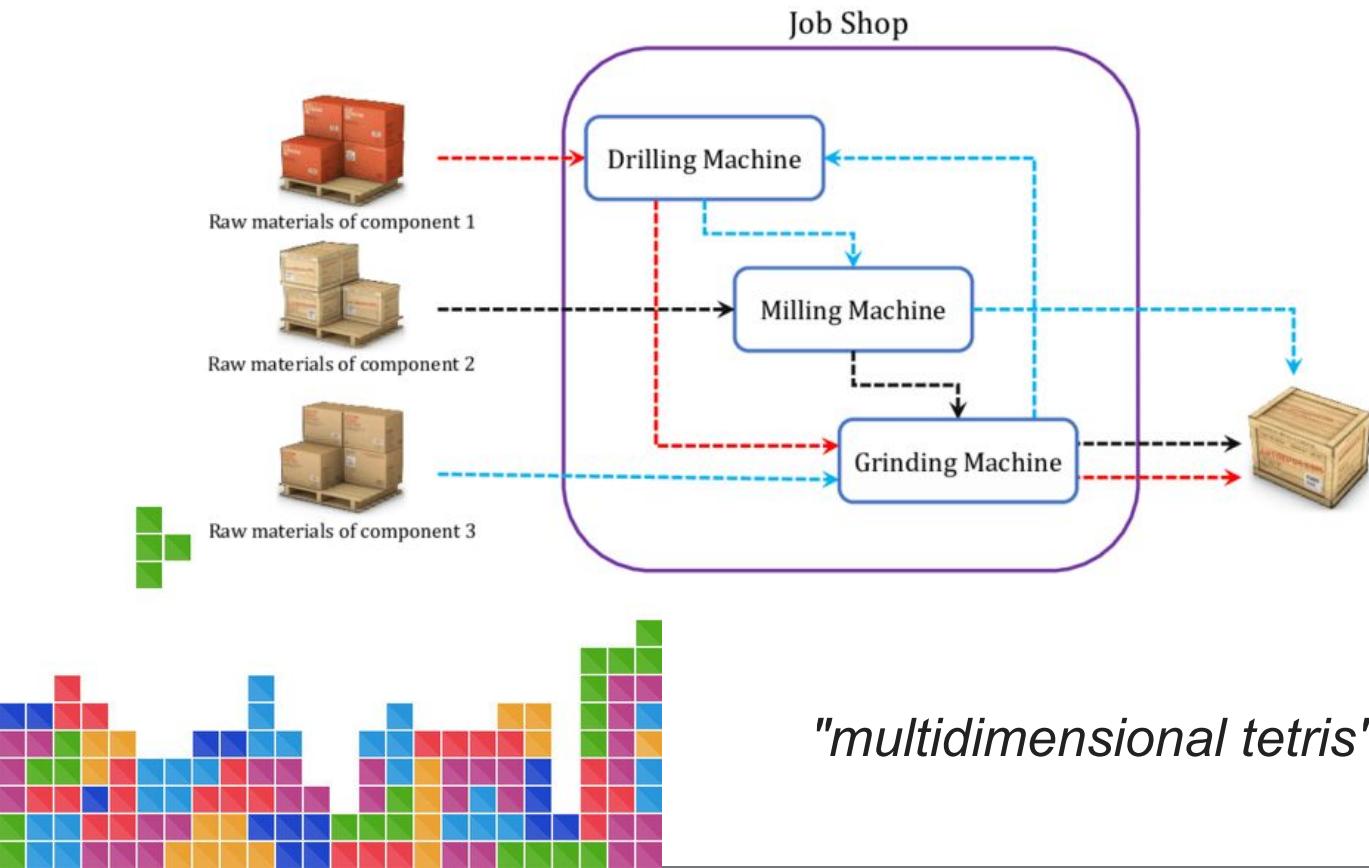
What is a Batch Job Scheduler?

A **batch job scheduler** is software that controls and tracks where and when batch jobs submitted to a computing system will eventually run on its compute resources.



Job-Shop Scheduling Problem

In the general job-shop scheduling problem, given N jobs of varying processing times, which need to be scheduled on M machines, each of which may have different processing powers, minimize the makespan – the total length of the schedule to complete all N jobs.



Batch Job Scheduling in Practice

In practice, real-world batch job schedulers typically have three primary aims when they are scheduling your jobs on the compute resources they help manage:

- To minimize the time between the job submission and job completion: no user job should stay pending in the queue for extensive periods of time
- To optimize compute resource utilization: no compute resource should be idle for extensive periods of time
- To maximize job throughput: process as many jobs as soon as possible

In most cases, the batch job schedulers on the HPC systems you will encounter typically also employ some form of what is known as a [fairshare scheduling](#) algorithm to match user jobs to compute nodes over time.

** All interactive jobs you run on an HPC system are usually treated the same as batch jobs.*

Table of Contents

- High-Performance Computing and System Architecture
- Batch Job Scheduling and Resource Management
- **Getting Started with the Slurm Workload Manager**
 - `squeue` - view information about jobs located in the Slurm scheduling queue
 - `sinfo` - view information about Slurm nodes and partitions
- First Batch Job(s) with Slurm
 - `sbatch` - submit a batch script to Slurm
 - `scancel` - used to signal or cancel jobs, job arrays or job steps
 - `sacct` - displays accounting data for all jobs in the Slurm database
- Best Practices with Slurm
- Questions & Answers (30 min)

Slurm Workload Manager

The [Slurm Workload Manager](#) is a free and open-source distributed batch job scheduler and resource manager for Unix-like operating systems used by the majority of the world's supercomputers and computer clusters today.

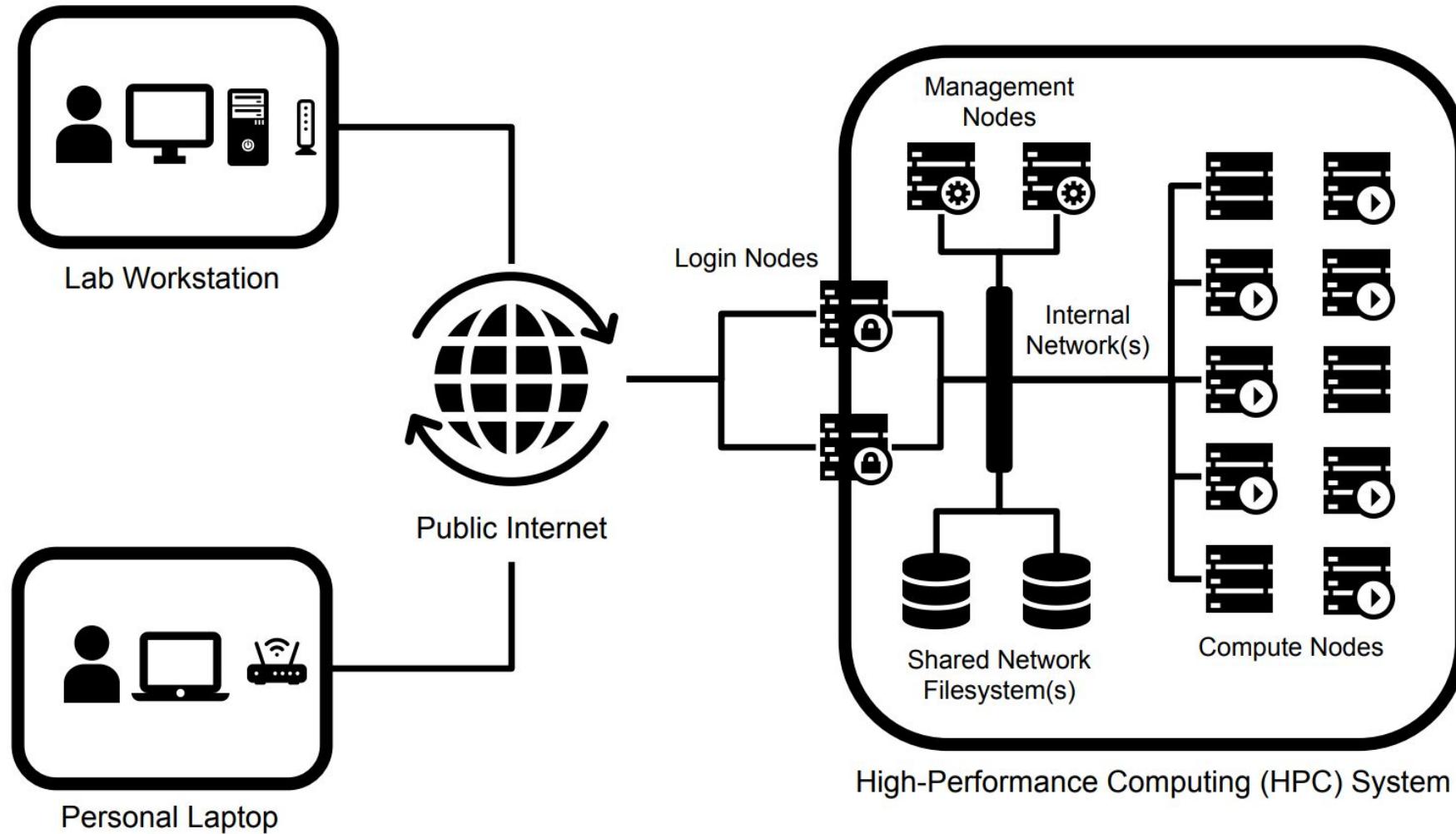
Slurm provides three key functions:

- the allocation of compute nodes for some period of time to process user jobs;
- a framework for starting, executing, and monitoring the progress of user jobs across a set of allocated compute nodes; and
- arbitrating the demand for compute nodes by managing a queue of pending user jobs

For more information, see the Slurm documentation: <https://slurm.schedmd.com>



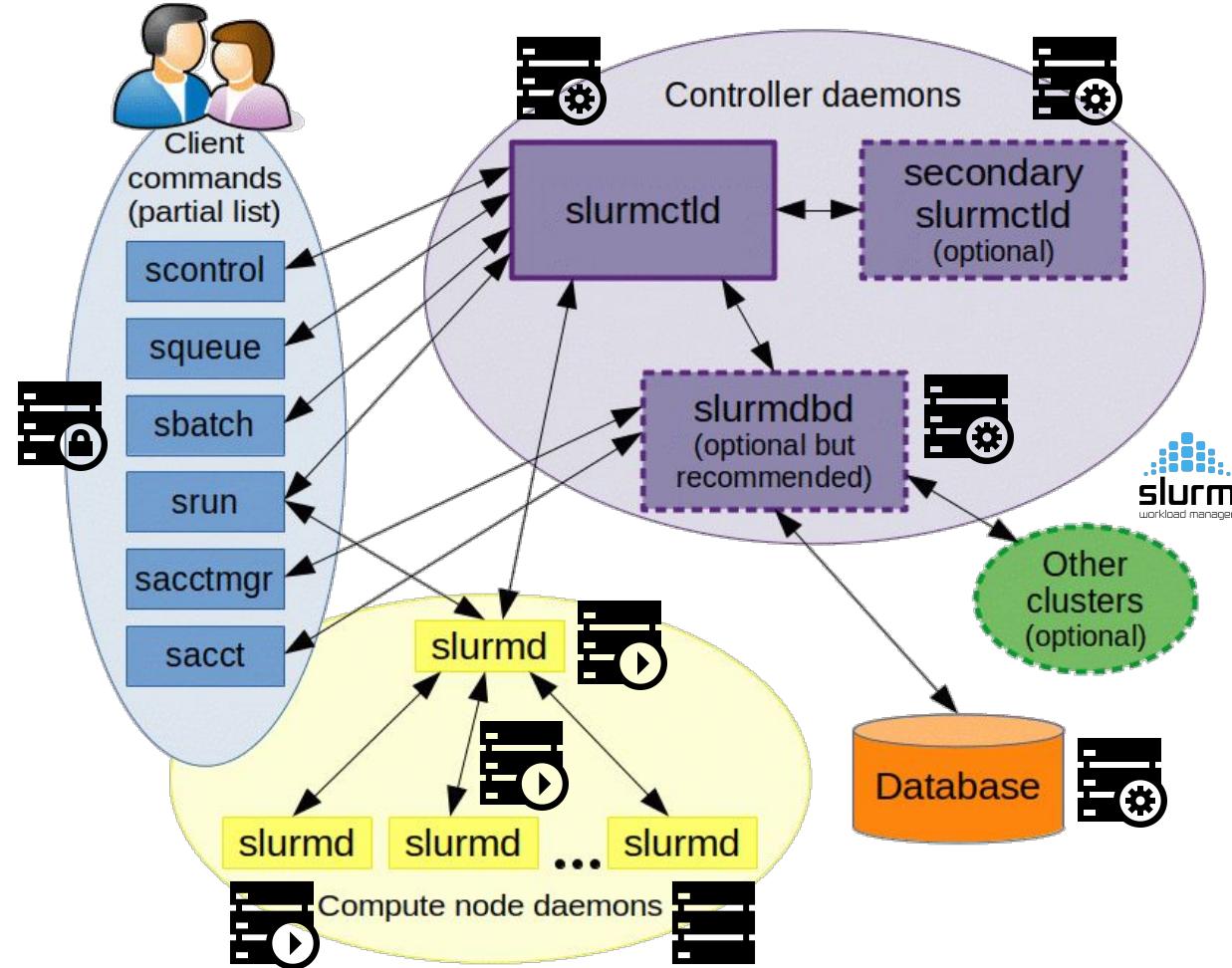
HPC System Architecture: Conceptual Model



Slurm Architecture

All Slurm schedulers are comprised of the three following types of service daemons:

- **slurmd** is the compute node daemon; it monitors all jobs running on the compute node, accepts jobs, launches jobs, and stops running jobs
- **slurmctld** is the central management daemon; it monitors all other Slurm daemons and resources, accepts jobs, and allocates resources to the jobs.
- **slurmdbd** is the accounting database daemon; it provides an interface for archiving accounting records to an external database



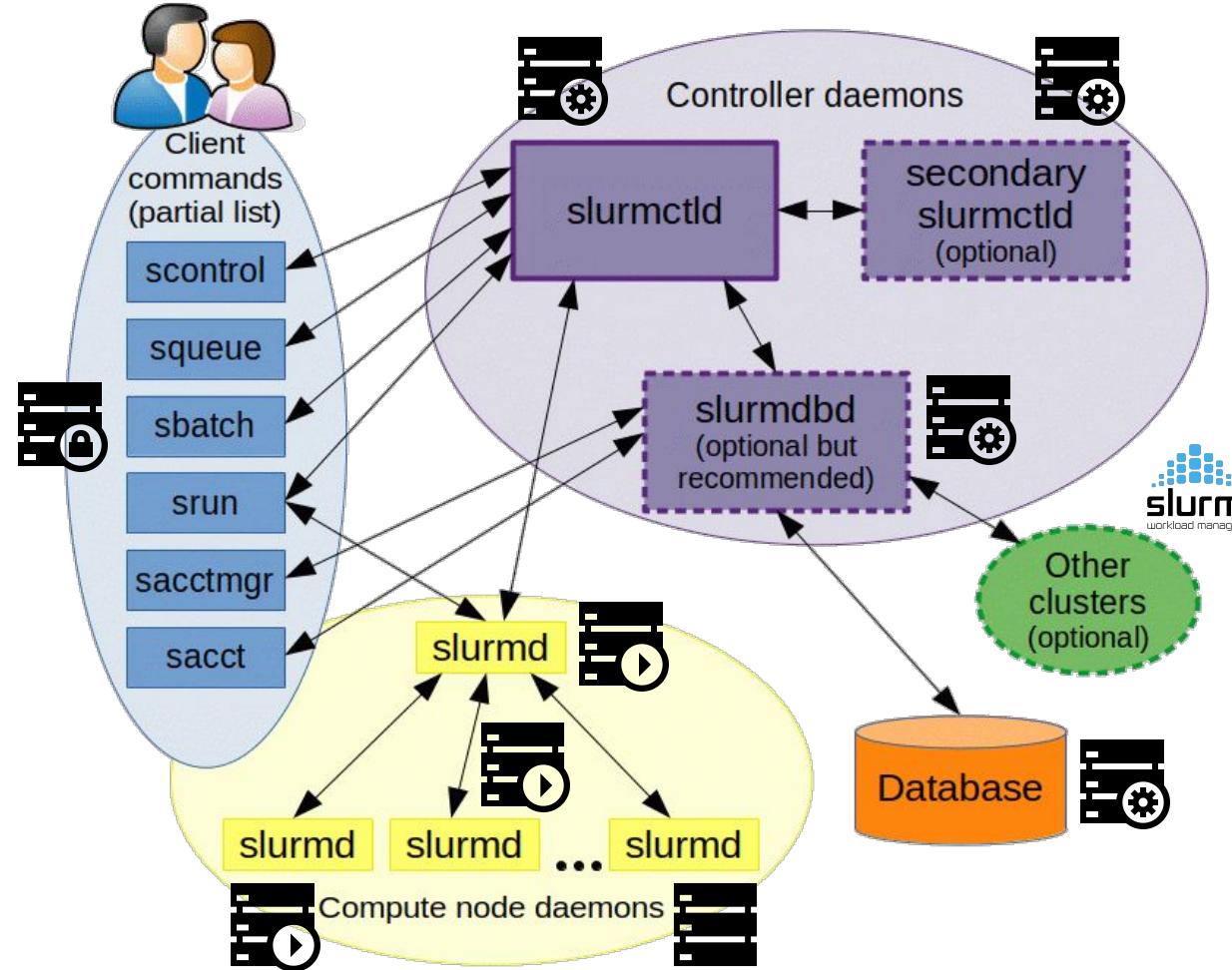
<https://slurm.schedmd.com/quickstart.html>

Slurm User Commands

The essential user commands you'll want to learn how to use when first getting started with Slurm are the following ones:

- [squeue](#)
- [sinfo](#)
- [sbatch](#)
- [scancel](#)
- [sacct](#)

In general, you will only run these commands interactively on an HPC system's login nodes.



<https://slurm.schedmd.com/quickstart.html>

squeue

The [**squeue**](#) command is used to query and view information about the user jobs submitted to the Slurm scheduling queue.

```
[mkandes@login01 ~]$ squeue
  JOBID PARTITION   NAME   USER ST      TIME NODES NODELIST(REASON)
28825741  compute touch fzzhao PD    0:00     1 (MaxMemPerLimit)
29153659  compute P3S3 dc1993 PD    0:00     1 (MaxMemPerLimit)
29153657  compute P3S3 dc1993 PD    0:00     1 (MaxMemPerLimit)
29184587  compute V0223 CT avibho PD  0:00     2 (MaxMemPerLimit)
28970190  compute Rxn7 saleh35 PD   0:00     5 (MaxMemPerLimit)
28970185  compute Rxn5 saleh35 PD   0:00     5 (MaxMemPerLimit)
28970184  compute Rxn4 saleh35 PD   0:00     5 (MaxMemPerLimit)
28970183  compute Mo-Pd saleh35 PD  0:00     5 (MaxMemPerLimit)
28970181  compute Rxn2 saleh35 PD   0:00     5 (MaxMemPerLimit)
29300918  compute V1_sim chenziao PD 0:00     1 (MaxMemPerLimit)
29372691  compute 18-10-12 vdommes PD 0:00    10 (Resources)
29372688  compute 18-6-128 vdommes PD 0:00     6 (Priority)
29372690  compute 18-6-128 vdommes PD 0:00     6 (Priority)
29372756  compute ceMvp-0 aikeliu PD 0:00     5 (Priority)
29373676  compute bs_fe prakash1 PD 0:00     2 (Priority)
29374534  compute A6007220 us3 PD    0:00     1 (Priority)
29373711  compute bs_fe prakash1 PD 0:00     2 (Priority)
29373974  compute A3aa oohiro PD    0:00     2 (Priority)
29373975  compute A3ab oohiro PD    0:00     2 (Priority)
29374059  compute A3ba oohiro PD    0:00     2 (Priority)
29374061  compute A3bb oohiro PD    0:00     2 (Priority)
29374083  compute A1c oohiro PD    0:00     2 (Priority)
29374940  compute 1.15.csh amy44ric PD 0:00     1 (Priority)
29374717  compute n32-t1-m jackyan1 PD 0:00    32 (Priority)
29374715  compute n32-t1-l jackyan1 PD 0:00    32 (Priority)
29374716  compute n32-t1-m jackyan1 PD 0:00    32 (Priority)
29374727  compute n32-t1-l jackyan1 PD 0:00    32 (Priority)
29374546  compute SHIRLEY_sjammuch PD 0:00     2 (Priority)
29373790  compute CHO_CHfs yuting82 PD 0:00     2 (Priority)
29374600  compute cp2k-com wborrell PD 0:00     1 (Priority)
29374602  compute cp2k-com wborrell PD 0:00     1 (Priority)
29375307  compute 1.15.csh amy44ric PD 0:00     1 (Priority)
29375309  compute 14.1.15. amy44ric PD 0:00     1 (Priority)
29374044  compute vasp6-co zwang21 PD 0:00     3 (Priority)
```

In general, you should only need to check the status of the jobs you've submitted to Slurm.

Recommendation: Create an alias in your shell's configuration file to only check the status of your jobs.

```
alias squeue="squeue -u ${USER}"
```

Do **NOT** run the squeue command incessantly, either on purpose or on accident.



slurm
workload manager

sinfo

The **sinfo** command is used to query and view information about the compute nodes and partitions managed by Slurm.

```
[mkandes@login01 ~]$ sinfo
PARTITION      AVAIL  TIMELIMIT  NODES  STATE NODELIST
compute        up     2:00:00:00    1  inval exp-14-53
compute        up     2:00:00:00    6  plnd exp-6-27,exp-8-[06-07,19],exp-9-[39-40]
compute        up     2:00:00:00    1  drng@ exp-6-30
compute        up     2:00:00:00    5  drain$ exp-1-38,exp-4-35,exp-8-10,exp-13-28,exp-14-30
compute        up     2:00:00:00    1  alloc$ exp-8-12
compute        up     2:00:00:00    4  down* exp-3-[25-28]
compute        up     2:00:00:00    1  drain exp-14-52
compute        up     2:00:00:00   545  alloc exp-1-[04-05,07-08,13-14,16-17,21-25,27,30,32,34,36,39,41-54,56],exp-2-[01
,04-09,11-14,16,19-28,30,32,34,36-37,39-40,43-46,48-52,54,56],exp-3-[01,03,06-07,09,11-19,21-23,29-30,33-35,37-40,42,44,
46-48,50-52,55-56],exp-4-[02-09,12-13,16-17,19-27,29-33,36,38-39,41-42,44-50,53-56],exp-5-[02-03,05,07-15,17-22,24,27-29
,31-37,39-40,42-43,46-50,55-56],exp-6-[02-05,07-11,13-17,20-26,28-29,34-51,53-56],exp-7-[02-12,14-15,17-25,27-35,43-44,5
0,52,54-56],exp-8-[04-05,08,11,13-18,20-33,36,38-44,46-49,51-56],exp-9-[01-03,06-18,20-23,27,29-30,32-38,41-42,44-54],ex
p-10-[01-11,15-17,19-47,49-56],exp-12-[01-33,38,41-49,52-56],exp-13-[01-02,04-12,14-22,24-27,29,31-44,51-52,54],exp-14-[0
1-13,17-18,20-22,25-29,31-35,37-39,41-48,50-51,54-56]
compute        up     2:00:00:00    46  resv exp-3-[20,43],exp-4-[15,37],exp-5-38,exp-6-[18,32-33],exp-7-[36-41,45-49]
,exp-8-34,exp-9-[05,25,43],exp-12-[34-37,51],exp-13-[03,45-50,55-56],exp-14-40,exp-16-[53-56],exp-17-[53-56]
compute        up     2:00:00:00   124  mix exp-1-[01-03,06,09-12,15,18-20,26,28-29,31,33,35,37,40,55],exp-2-[02-03,10
,15,17-18,29,31,33,35,38,41-42,47,53,55],exp-3-[02,04-05,08,10,24,31-32,36,41,45,49,53-54],exp-4-[01,10-11,14,18,28,34,4
,04,43,51-52],exp-5-[01,04,06,16,23,25-26,30,41,44-45,51-54],exp-6-[01,06,12,19,31,52],exp-7-[01,13,16,26,42,51,53],exp-8
-[01-03,09,35,37,45,50],exp-9-[04,19,24,26,28,31],exp-10-[12-14,18,48],exp-12-[39-40,50],exp-13-[13,23,30,53],exp-14-[14
,16,19,23-24,36,49]
cw3e-compute   up     2:00:00:00    5   plnd exp-18-[46-47,55-56,60]
cw3e-compute   up     2:00:00:00    1  drng@ exp-18-01
cw3e-compute   up     2:00:00:00    1  drain$ exp-18-45
cw3e-compute   up     2:00:00:00    5   mix exp-18-[12-13,17,48,58]
cw3e-compute   up     2:00:00:00   48  alloc exp-18-[02-11,14-16,18-44,49-54,57,59]
cw3e-shared     up     2:00:00:00    5   plnd exp-18-[46-47,55-56,60]
cw3e-shared     up     2:00:00:00    1  drng@ exp-18-01
cw3e-shared     up     2:00:00:00    1  drain$ exp-18-45
cw3e-shared     up     2:00:00:00    5   mix exp-18-[12-13,17,48,58]
cw3e-shared     up     2:00:00:00   48  alloc exp-18-[02-11,14-16,18-44,49-54,57,59]
debug          up     30:00    1   mix exp-9-55
debug          up     30:00    1   idle exp-9-56
gpu            up     2:00:00:00    1  drain$ exp-8-60
```

Slurm partitions are used to group nodes into logical, sometimes overlapping sets of nodes.

Each partition may have a unique set of constraints and features such as job size and runtime limits, types of resources available, user-based access permissions, etc.

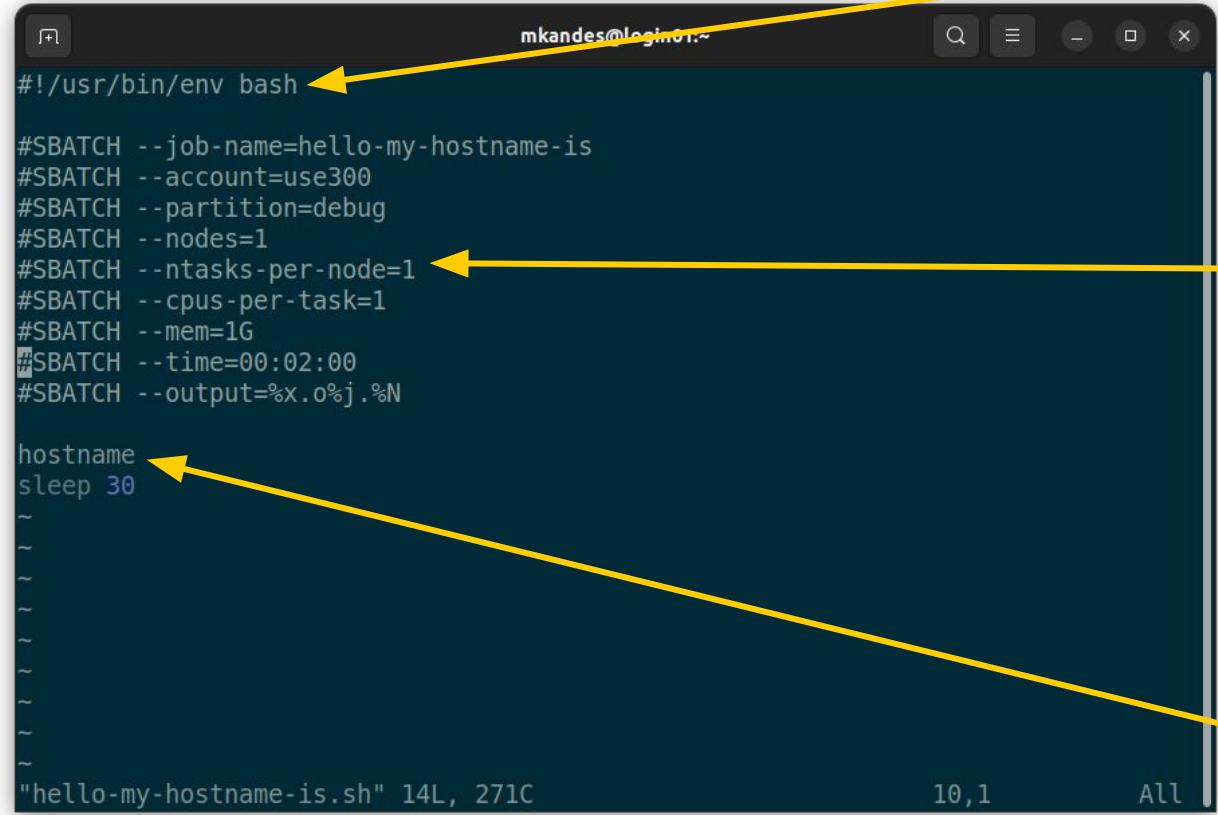
You should always select the most appropriate partition for any job.

https://www.sdsc.edu/support/user_guides/expanse.html#running

Table of Contents

- High-Performance Computing and System Architecture
- Batch Job Scheduling and Resource Management
- Getting Started with the Slurm Workload Manager
 - `squeue` - view information about jobs located in the Slurm scheduling queue
 - `sinfo` - view information about Slurm nodes and partitions
- First Batch Job(s) with Slurm
 - `sbatch` - submit a batch script to Slurm
 - `scancel` - used to signal or cancel jobs, job arrays or job steps
 - `sacct` - displays accounting data for all jobs in the Slurm database
- Best Practices with Slurm
- Questions & Answers (30 min)

Exercise 1: Hello, my hostname is



A screenshot of a terminal window titled 'mkandes@logi-OptiPlex-5090:~'. The window contains a bash script with the following content:

```
#!/usr/bin/env bash

#SBATCH --job-name=hello-my-hostname-is
#SBATCH --account=use300
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=1G
#SBATCH --time=00:02:00
#SBATCH --output=%x.%o%j.%N

hostname
sleep 30

~
```

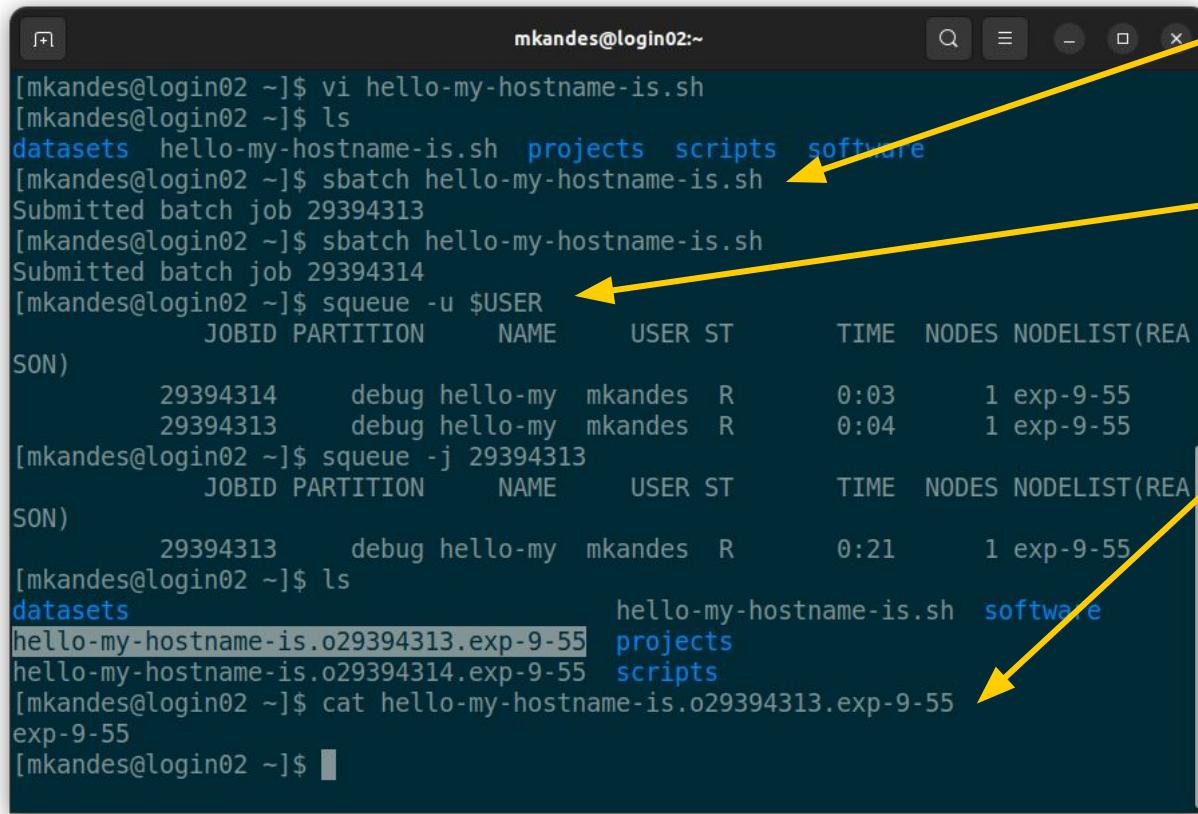
The terminal status bar at the bottom shows the file name "hello-my-hostname-is.sh" and its size "14L, 271C". It also displays the current cursor position "10,1" and the word "All".

- shebang (#!) character sequence is used to specify an interpreter that will execute the commands present in a plain text file.
- #SBATCH directives are used to specify the computational resource requirements for your job and various other attributes of it. e.g., name of the job and output files, which account to charge the job to, etc.
- All executable commands to be run must follow the list of #SBATCH directives specified in any job script.

Exercise materials are available at
<https://github.com/sdsc-complecs/batch-computing>

sbatch

The **sbatch** command is used to submit a batch job script to Slurm.



A terminal window titled "mkandes@login02:~" showing the following session:

```
[mkandes@login02 ~]$ vi hello-my-hostname-is.sh
[mkandes@login02 ~]$ ls
datasets  hello-my-hostname-is.sh  projects  scripts  software
[mkandes@login02 ~]$ sbatch hello-my-hostname-is.sh
Submitted batch job 29394313
[mkandes@login02 ~]$ sbatch hello-my-hostname-is.sh
Submitted batch job 29394314
[mkandes@login02 ~]$ squeue -u $USER
  JOBID PARTITION      NAME      USER   ST      TIME   NODES NODELIST(REA
SON)
  29394314    debug hello-my  mkandes   R      0:03      1 exp-9-55
  29394313    debug hello-my  mkandes   R      0:04      1 exp-9-55
[mkandes@login02 ~]$ squeue -j 29394313
  JOBID PARTITION      NAME      USER   ST      TIME   NODES NODELIST(REA
SON)
  29394313    debug hello-my  mkandes   R      0:21      1 exp-9-55
[mkandes@login02 ~]$ ls
datasets          hello-my-hostname-is.sh  projects
                  hello-my-hostname-is.o29394313.exp-9-55  scripts
                  hello-my-hostname-is.o29394314.exp-9-55
[mkandes@login02 ~]$ cat hello-my-hostname-is.o29394313.exp-9-55
exp-9-55
[mkandes@login02 ~]$
```

- Upon submission, every batch job is assigned a SLURM_JOB_ID number.
 - After submission, you can monitor the status of your jobs with the squeue command.
 - While a job runs, it will write standard output (and error) data to the file(s) named in your #SBATCH directives.
- #SBATCH --output=%x.o%j.%N
- %x = SLURM_JOB_NAME
%j = SLURM_JOB_ID
%N = hostname -s

Exercise 2: Roll 4 Pi



The screenshot shows a terminal window titled "mkandes@login01:~". The window contains a shell script named "roll-4-pi.sh". The script starts with a shebang "#!/usr/bin/env bash" and several "#SBATCH" directives for job submission. It then uses "module" commands to load software modules: "module reset", "module load gcc/10.2.0", "module load python/3.8.12", "module list", and "printenv". Finally, it runs a Python script "pi.py" with the command "time -p python3 pi.py 10000000000". The terminal status bar at the bottom indicates the file is "roll-4-pi.sh" with 19L and 359C.

```
#!/usr/bin/env bash

#SBATCH --job-name=roll-4-pi
#SBATCH --account=use300
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=1G
#SBATCH --time=00:02:00
#SBATCH --output=%x.%0%j.%N

module reset
module load gcc/10.2.0
module load python/3.8.12
module list
printenv

time -p python3 pi.py 10000000000

"roll-4-pi.sh" 19L, 359C
```

Exercise materials are available at
<https://github.com/sdsc-complecs/batch-computing>

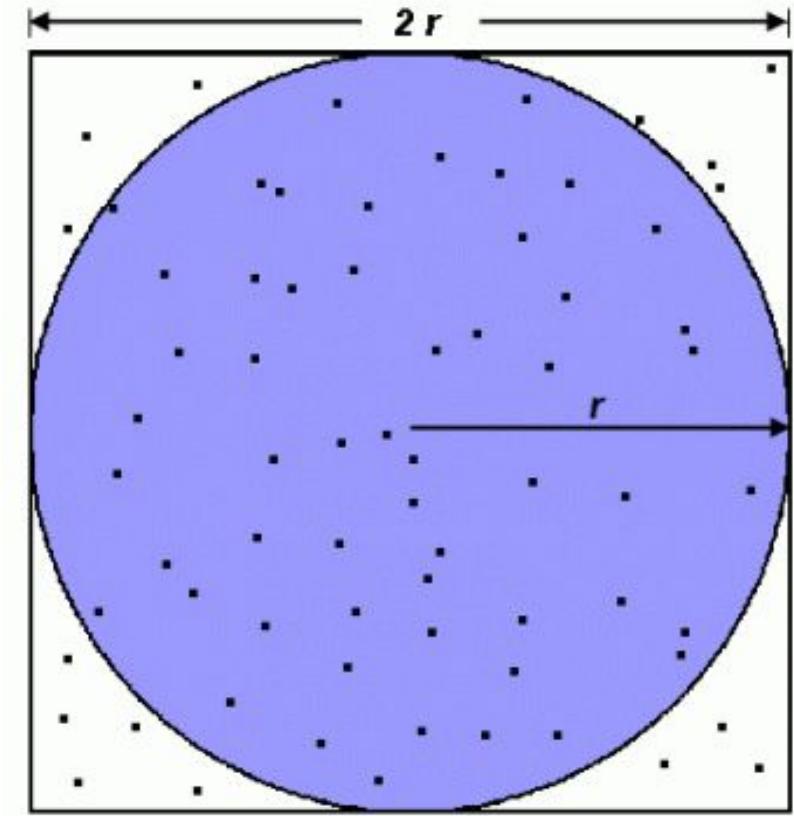
In most cases, you will want to follow the #SBATCH directives in your job script with a set of commands that configure the software environment in which the primary executable will run.

On many HPC systems, you will configure the software environment using pre-compiled and installed software modules.

Here, we estimate the value of Pi using a Monte Carlo method implemented in Python. The prepended **time** command measures the runtime of the estimation.

sbatch and roll (100M samples)

```
mkandes@login02 ~]$ ls
datasets          hello-my-hostname-is.sh  roll-4-pi.sh
hello-my-hostname-is.o29394313.exp-9-55  pi.py      scripts
hello-my-hostname-is.o29394314.exp-9-55  projects   software
[mkandes@login02 ~]$ sinfo -p debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug       up    30:00        1  free  exp-9-56
debug       up    30:00        1  free  exp-9-55
[mkandes@login02 ~]$ sbatch roll-4-pi.sh
Submitted batch job 29402750
[mkandes@login02 ~]$ squeue -u $USER
             JOBID PARTITION     NAME     USER ST      TIME  NODES NODELIST(REA
SON)
 29402750    debug roll-4-pi mkandes R      0:22      1 exp-9-55
[mkandes@login02 ~]$ squeue -u $USER
             JOBID PARTITION     NAME     USER ST      TIME  NODES NODELIST(REA
SON)
[mkandes@login02 ~]$ tail -n 4 roll-4-pi.o29402750.exp-9-55
3.141592653589793
real 62.54
user 62.36
sys 0.01
[mkandes@login02 ~]$
```



$$A_S = (2r)^2 = 4r^2$$

$$A_C = \pi r^2$$

$$\pi = 4 \times \frac{A_C}{A_S}$$

What do you expect will happen if we increase the number of samples to one billion and resubmit the job??

sacct

The sacct command displays job data from the Slurm accounting database.

```
mkanedes@login02:~
```

```
hello-my-hostname-is.o29394314.exp-9-55 roll-4-pi.o29402750.exp-9-55
hello-my-hostname-is.sh roll-4-pi.sh
[mkandes@login02 ~]$ sinfo -p debug
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
debug up 30:00 2 idle exp-9-[55-56]
[mkandes@login02 ~]$ sbatch roll-4-pi.sh
Submitted batch job 29408787
[mkandes@login02 ~]$ squeue -u $USER
          JOBID PARTITION     NAME     USER ST      TIME  NODES NODELIST(REA
SON)
          29408787     debug roll-4-pi mkandes R      1:19      1 exp-9-55
[mkandes@login02 ~]$ tail -n 4 roll-4-pi.o29408787.exp-9-55
BASH_FUNC_ml%%=() { eval $($LMOD_DIR/ml_cmd "$@")
}
_=#!/usr/bin/printenv
slurmstepd: error: *** JOB 29408787 ON exp-9-55 CANCELLED AT 2024-03-20T18:12:28
  DUE TO TIME LIMIT ***
[mkandes@login02 ~]$ sacct -j 29408787
JobID      JobName Partition Account AllocCPUS      State ExitCode
-----  -----
29408787    roll-4-pi     debug   use300        1  TIMEOUT       0:0
29408787.ba+    batch     debug   use300        1  CANCELLED    0:15
29408787.ex+    extern     debug   use300        1  COMPLETED    0:0
[mkandes@login02 ~]$
```

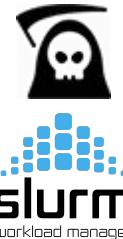
What can you do when you submit a job you later realize has a mistake and/or might fail?

Use to look up information about completed and/or failed jobs when they are no longer visible in `squeue`.

Quickly check the final state of a job and answer these questions: Did the job run and exit normally? If not, does Slurm know why it failed?

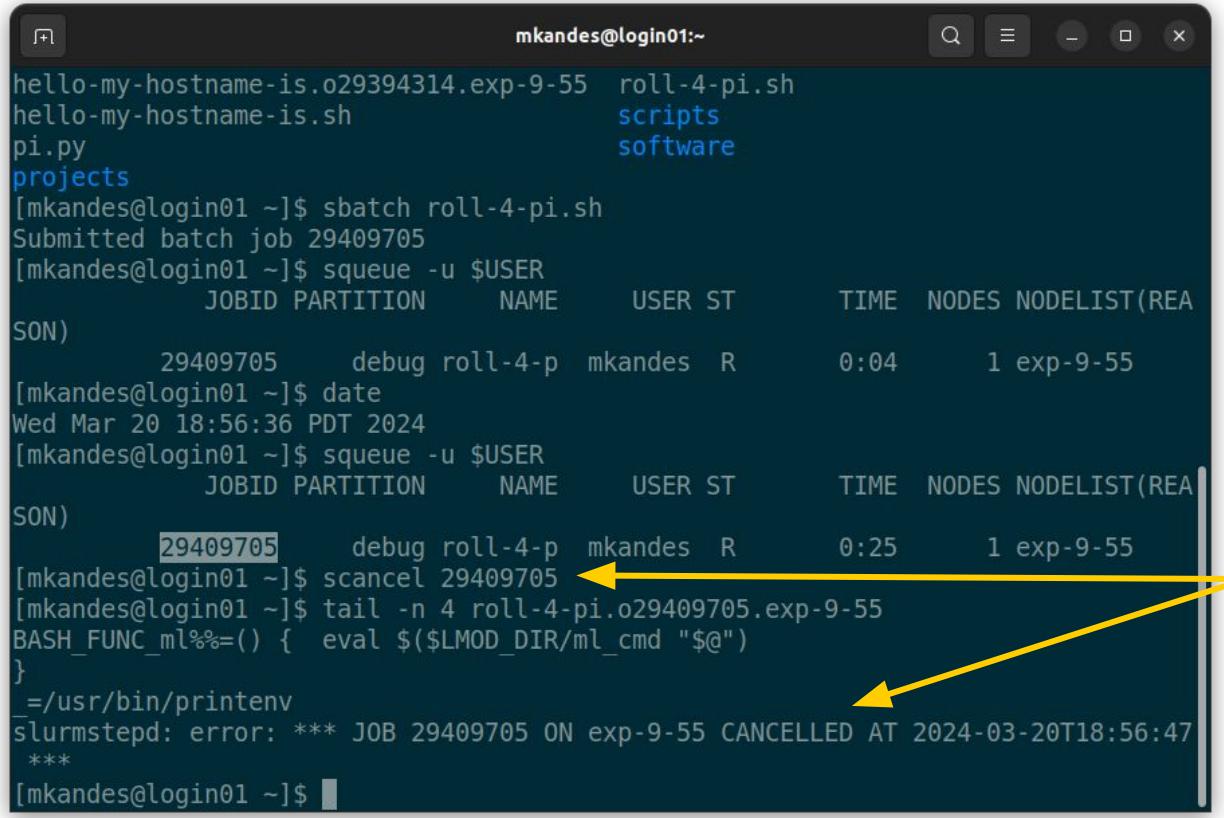
One of the most common failure modes for any batch job on an HPC system is running out of time.

Do **NOT** run extensive `sacct` queries without good reason.



scancel

The [scancel](#) command is used to signal or cancel jobs, job arrays or job steps.



A terminal window titled "mkandes@login01:~". The session shows:

```
hello-my-hostname-is.o29394314.exp-9-55 roll-4-pi.sh
hello-my-hostname-is.sh           scripts
pi.py                           software
projects

[mkandes@login01 ~]$ sbatch roll-4-pi.sh
Submitted batch job 29409705
[mkandes@login01 ~]$ squeue -u $USER
      JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REA
SON)
      29409705      debug roll-4-p  mkandes R      0:04      1 exp-9-55
[mkandes@login01 ~]$ date
Wed Mar 20 18:56:36 PDT 2024
[mkandes@login01 ~]$ squeue -u $USER
      JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REA
SON)
      29409705      debug roll-4-p  mkandes R      0:25      1 exp-9-55
[mkandes@login01 ~]$ scancel 29409705
[mkandes@login01 ~]$ tail -n 4 roll-4-pi.o29409705.exp-9-55
BASH_FUNC_ml%%=() { eval $($LMOD_DIR/ml_cmd "$@")
}
/usr/bin/printenv
slurmstepd: error: *** JOB 29409705 ON exp-9-55 CANCELLED AT 2024-03-20T18:56:47
 ***
***
```

Two yellow arrows point from the text "Use whenever you want to remove a pending job from the queue or stop a running job." to the command "scancel 29409705" and its output.

Use whenever you want to remove a pending job from the queue or stop a running job.

Table of Contents

- High-Performance Computing and System Architecture
- Batch Job Scheduling and Resource Management
- Getting Started with the Slurm Workload Manager
 - `squeue` - view information about jobs located in the Slurm scheduling queue
 - `sinfo` - view information about Slurm nodes and partitions
- First Batch Job(s) with Slurm
 - `sbatch` - submit a batch script to Slurm
 - `scancel` - used to signal or cancel jobs, job arrays or job steps
 - `sacct` - displays accounting data for all jobs in the Slurm database
- Best Practices with Slurm
- Questions & Answers (30 min)

Parallel Computing Myth #2: Scheduler Variant

Simply requesting more compute resources from your batch job scheduler will automatically reduce the time to solution for your problem.

WRONG!

The truth is your application must be designed to take advantage of the additional compute resources you've requested. If it is designed to do so, then you generally must still configure your batch job to correctly utilize those resources.



Prerequisite Knowledge:

- [Parallel Computing Concepts](#)

Exercise 3: Build Job, Run Job

If you need to compile and install software on an HPC system, then you should consider writing a batch job to build and install the software instead of proceeding to do so interactively from the command-line.

There are a number of advantages of using this approach:

- the build job script will document how to build and install the software
- the build and install logs will be available in the job's standard output (and/or error) files
- the software may need to be compiled on a specific compute node architecture
- the software environment created within the build job to compile the software will need to be replicated in the job scripts that eventually run the software anyway

Exercise materials are available at
<https://github.com/sdsc-complecs/batch-computing>

Parallel Pi with OpenMP

Parallel computing jobs can take advantage of more CPU resources

```
mkandes@login01:~
```

```
#!/usr/bin/env bash

#SBATCH --job-name=build-pi-omp
#SBATCH --account=use300
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=1G
#SBATCH --time=00:02:00
#SBATCH --output=%x.o%j.%N

module reset
module load gcc/10.2.0
module list
printenv

make
~

"build-pi-omp.sh" 18L, 308C          18,1      All
```

Build Job

```
mkandes@login01:~
```

```
#!/usr/bin/env bash

#SBATCH --job-name=run-pi-omp
#SBATCH --account=use300
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=4
#SBATCH --mem=1G
#SBATCH --time=00:02:00
#SBATCH --output=%x.o%j.%N

module reset
module load gcc/10.2.0
module list
export OMP_NUM_THREADS="${SLURM_CPUS_PER_TASK}"
printenv

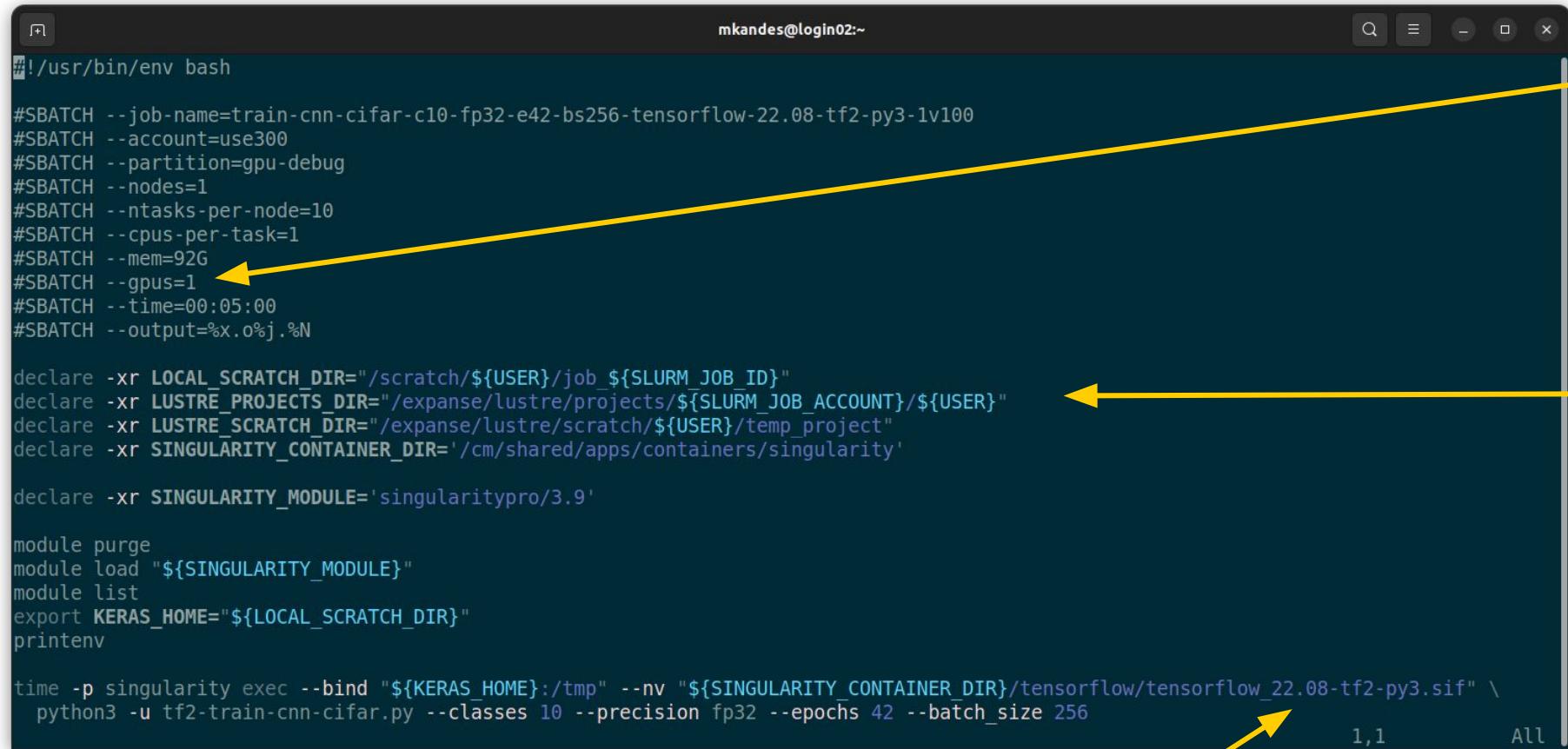
time -p ./pi_omp.x -s 10000000000
~

"run-pi-omp.sh" 19L, 383C          3,1      All
```

Run Job

Use Slurm environment variables!

Exercise 4: GPU-Accelerated Workloads



```
#!/usr/bin/env bash

#SBATCH --job-name=train-cnn-cifar-c10-fp32-e42-bs256-tensorflow-22.08-tf2-py3-1v100
#SBATCH --account=use300
#SBATCH --partition=gpu-debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=10
#SBATCH --cpus-per-task=1
#SBATCH --mem=92G
#SBATCH --gpus=1
#SBATCH --time=00:05:00
#SBATCH --output=%x.%o%j.%N

declare -xr LOCAL_SCRATCH_DIR="/scratch/${USER}/job_${SLURM_JOB_ID}"
declare -xr LUSTRE_PROJECTS_DIR="/expanses/lustre/projects/${SLURM_JOB_ACCOUNT}/${USER}"
declare -xr LUSTRE_SCRATCH_DIR="/expanses/lustre/scratch/${USER}/temp_project"
declare -xr SINGULARITY_CONTAINER_DIR='/cm/shared/apps/containers/singularity'

declare -xr SINGULARITY_MODULE='singularitypro/3.9'

module purge
module load "${SINGULARITY_MODULE}"
module list
export KERAS_HOME="${LOCAL_SCRATCH_DIR}"
printenv

time -p singularity exec --bind "${KERAS_HOME}:/tmp" --nv "${SINGULARITY_CONTAINER_DIR}/tensorflow/tensorflow_22.08-tf2-py3.sif" \
    python3 -u tf2-train-cnn-cifar.py --classes 10 --precision fp32 --epochs 42 --batch_size 256
```

GPUs are now a first-class compute resource in Slurm

Define your own environment variables as needed!

Exercise materials are available at
<https://github.com/sdsc-complecs/batch-computing>

Your job's software environment can be a mix of standard HPC software modules and alternative software package management methods. e.g., Singularity containers, conda environments

Exercise 5: Multinode MPI Jobs

```
#!/usr/bin/env bash

#SBATCH --job-name=gromacs-2020.4-2ufeq67-aocc-3.2.0-io3s466-openmpi-4.1.3-xigazqd-water-cut1.0_GMX50_bare-3072-4-node-128-mpi-1-omp
#SBATCH --account=use300
#SBATCH --partition=compute
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=128
#SBATCH --cpus-per-task=1
#SBATCH --mem=243G
#SBATCH --time=00:05:00
#SBATCH --output=%x.o%j.%N

declare -xr COMPILER_NAME='aocc'
declare -xr COMPILER_VERSION='3.2.0'
declare -xr COMPILER_VARIANT='io3s466'
declare -xr COMPILER_MODULE="${COMPILER_NAME}/$(lsb_release -c -s)/${COMPILER_VARIANT}"

declare -xr MPI_NAME='openmpi'
declare -xr MPI_VERSION='4.1.3'
declare -xr MPI_VARIANT='xigazqd'
declare -xr MPI_MODULE="${MPI_NAME}/$(lsb_release -c -s)/${MPI_VARIANT}"

declare -xr APPLICATION_NAME='gromacs'

mkandes@login01:~
```



```
mkandes@login01:~
```

```
if [[ ! -f "${SLURM_SUBMIT_DIR}/benchmarks/${GROMACS_BENCHMARK_TARBALL}" ]]; then
    wget "${GROMACS_ROOT_URL}/benchmarks/${GROMACS_BENCHMARK_TARBALL}"
fi
tar -xzvf "${GROMACS_BENCHMARK_TARBALL}"
fi

cd "${SLURM_SUBMIT_DIR}"

time -p mpirun -n 1 gmx_mpi grompp \
-f "${GROMACS_BENCHMARK_DATA_DIR}/pme.mdp" \
-c "${GROMACS_BENCHMARK_DATA_DIR}/conf.gro" \
-p "${GROMACS_BENCHMARK_DATA_DIR}/topol.top" \
-po "mdout.${SLURM_JOB_NAME}.${SLURM_JOB_ID}.mdp" \
-o "topol.${SLURM_JOB_NAME}.${SLURM_JOB_ID}.tpr"

time -p mpirun -n "${SLURM_NTASKS}" --bind-to core --map-by ppr:1:core:PE=1 \
--display-allocation --display-map --report-bindings --verbose gmx_mpi mdrun \
-nb cpu -pme cpu -bonded cpu -pin on -resethway -noconfout -nsteps 10000 \
-s "topol.${SLURM_JOB_NAME}.${SLURM_JOB_ID}.tpr" \
-cpo "state.${SLURM_JOB_NAME}.${SLURM_JOB_ID}.cpt" \
-e "ener.${SLURM_JOB_NAME}.${SLURM_JOB_ID}.edr" \
-g "md.${SLURM_JOB_NAME}.${SLURM_JOB_ID}.log" \
-v
```

You can request multiple compute nodes for your jobs. But only do so if the software you're running supports distributed memory parallelism. e.g., MPI

Exercise materials are available at
<https://github.com/sdsc-complexs/batch-computing>

But again, you will still likely need to configure the call to the application to use the requested resources correctly.

Some final notes

- When you get started on a new HPC system, or begin working with a new piece of software, you may not know computational requirements for the types of workloads you need to run. Experiment! Start small on number of CPUs/GPUs, amount of memory, then scale up your job resource requests as needed. Start long on runtime, then dial it back.
- Be aware of how your HPC system charges for the resources requested from the scheduler. Some charge by node-hour, some by CPU-core-hour, some by GPU-hour, some by memory, or even a combination of all of the above. In general, you are only charged for the resources consumed, not those requested.
- Read your HPC system's user guide, the Slurm documentation, and/or the documentation to the specific software you're attempting to run.
- If you have any questions, please don't hesitate to contact your HPC user services support team. They are there to help you understand how to utilize these specialized resources effectively for your research.

Scheduler Rosetta Stone

| Common User Commands | PBS/Torque | Slurm | LSF | SGE | LoadLeveler |
|-----------------------|--|---|-------------------------------|--------------------|--|
| Job submission | qsub [script_file] | sbatch [script_file] | bsub [script_file] | qsub [script_file] | llsubmit [script_file] |
| Job deletion | qdel [job_id] | scancel [job_id] | bkill [job_id] | qdel [job_id] | llcancel [job_id] |
| Job status | qstat [job_id] | squeue [job_id] | bjobs | qstat [-j job_id] | llq -u [username] |
| Job status (by user) | qstat -u [user_name] | squeue -u [user_name] | bjobs | qstat [-j job_id] | llq -u [user_name] |
| Job hold | qhold [job_id] | scontrol hold [job_id] | bstop [job_id] | qhold [job_id] | llhold -r [job_id] |
| Job release | qrfls [job_id] | scontrol release [job_id] | gresume [job_id] | qrsls [job_id] | llhold -r [job_id] |
| Queue list | qstat -Q | squeue | bqueues | qconf -sql | llclass |
| GUI | xpbsmon | sview | xlsf/xlsbatch | N/A | xload |
| Environment Variables | | | | | |
| Job ID | \$PBS_JOBID | \$\$SLURM_JOBID | \$_LSB_JOBID | \$JOB_ID | \$LOAD_STEP_ID |
| Working Directory | \$PBS_O_WORKDIR | Use "pwd" command | Use "pwd" command | \$\$SGE_O_WORKDIR | \$LOADL_STEP_INITDIR |
| Node List | \$PBS_NODEFILE | \$\$SLURM_JOB_NODELIST | \$_LSB_HOSTS/\$_LSB_MCPU_HOST | \$PE_HOSTFILE | \$LOADL_PROCESSOR_LIST |
| Job Specification | | | | | |
| Script directive | #PBS | #SBATCH | #BSUB | #\$ | #@ |
| Queue | -q [queue] | -p [queue] | -q [queue] | -q [queue] | class=[queue] |
| Node Count | -l nodes=[count] | -N [min max] | -n [count] | N/A | node=[count] |
| CPU Count | -l ppn=[count] | -n [count] | -n [count] | -pe ompi [#] | tasks_per_node=[count] |
| Wall Clock Limit | -l walltime=[hh:mm:ss] | -t [min] OR -t [days-hh:mm:ss] | -W | -l time=[hh:mm:ss] | wall_clock_limit=[hh:mm:ss] |
| Standard Output File | -o [file_name] | -o [file_name] | -o [file_name] | -o [path] | output=[file] |
| Standard Error File | -e [file_name] | e [file_name] | -e [file_name] | -e [path] | error=[file] |
| Combine stdout/err | -j oe (both to stdout) OR -j eo (both to stderr) | (use -o without -e) | (use -o without -e) | | |
| Copy Environment | -V | --export=[ALL NONE variables] | ? | -V | environment=COPY_ALL |
| Event Notification | -m abe | --mail-type=[events] | -B or -N | -m abe | notification=start error complete never always |
| Email Address | -M [address] | --mail-user=[address] | -u [address] | -M [address] | notify_user=[address] |
| Job Name | -N [name] | --job-name=[name] | -J [name] | -N [name] | job_name=[name] |
| Job Restart | -r [y n] | (use -o without -e) | -r | -r [yes no] | restart=[yes no] |
| Job Type | N/A | N/A | N/A | N/A | job_type=[type] |
| Working Directory | N/A | --workdir=[dir_name] | (submission directory) | -wd [directory] | initialdir=[directory] |
| Resource Sharing | -l naccesspolicy=singlejob | --exclusive OR--shared | -x | N/A | node_usage=not_shared |
| Memory Size | -l mem=[MB] | --mem=[MB] OR --mem-per-cpu=[MB] | -M [MB] | -mem [GB] | requirements=(Memory >= [MB]) |
| Project to charge | -W group_list=<project> | --account=[project] | -P <project> | | |
| Resource Requirements | | | -R | | |
| Job dependency | | --depend=[job_id:state] | -w [done exit finish] | | |
| Job project | | --wckey=[name] | -P [name] | | |
| Job host preference | | --nodelist=[nodes] AND/OR --exclude=[nodes] | -m [nodes] | | |

<https://slurm.schedmd.com/rosetta.html>

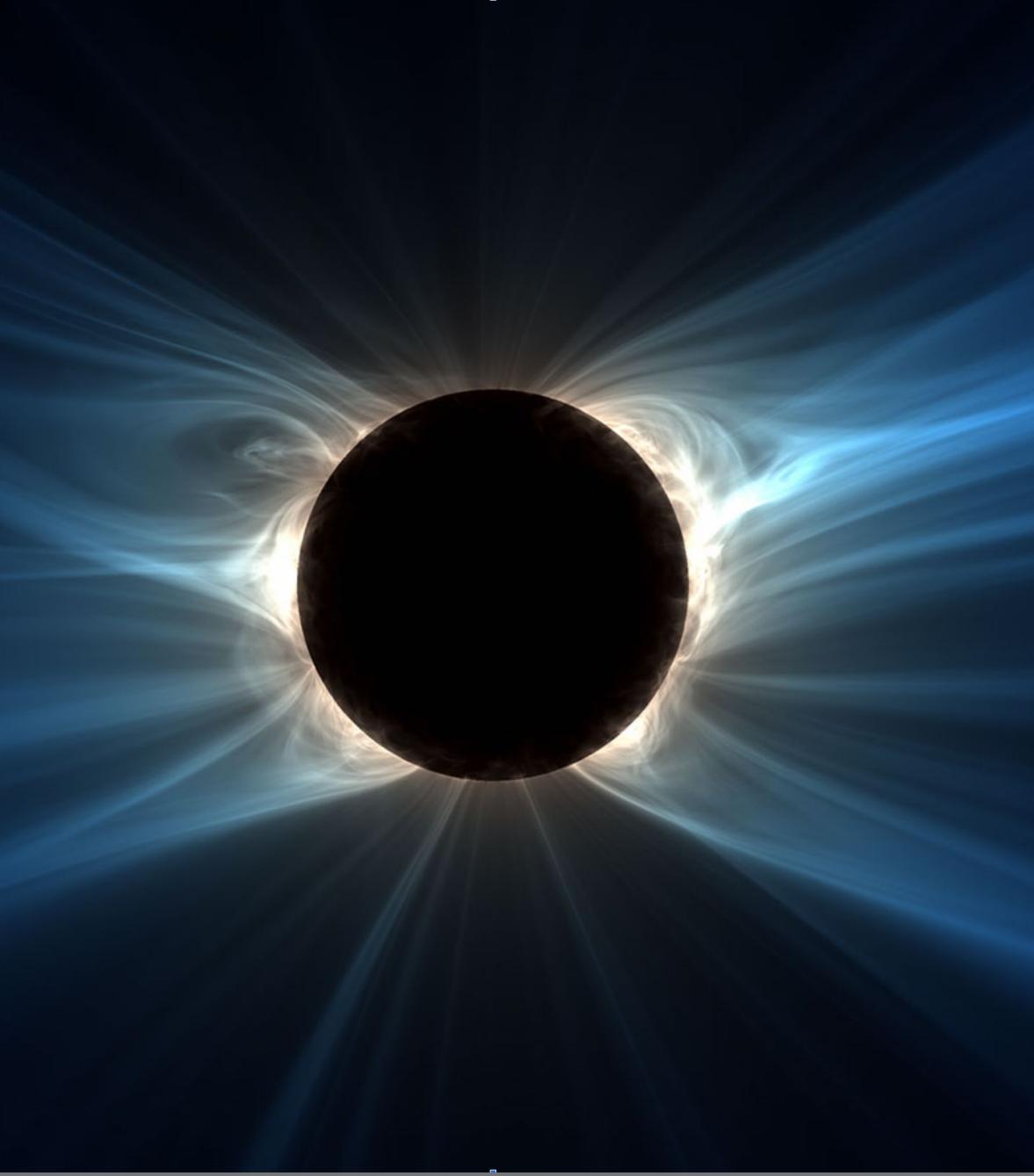
Additional Information

- [Slurm Quick Start User Guide](#)
- [HPCWIKI: Scheduling Basics](#)
- [Introduction to HPC](#)
- [HPC in a Day](#)
- [Google's Shell Style Guide](#)



Table of Contents

- High-Performance Computing and System Architecture
- Batch Job Scheduling and Resource Management
- Getting Started with the Slurm Workload Manager
 - `squeue` - view information about jobs located in the Slurm scheduling queue
 - `sinfo` - view information about Slurm nodes and partitions
- First Batch Job(s) with Slurm
 - `sbatch` - submit a batch script to Slurm
 - `scancel` - used to signal or cancel jobs, job arrays or job steps
 - `sacct` - displays accounting data for all jobs in the Slurm database
- Best Practices with Slurm
- Questions & Answers (30 min)



Questions & Answers