

Getting Started with Batch Job Scheduling: Slurm Edition

Marty Kandes, Ph.D.

Computational & Data Science Research Specialist
High-Performance Computing User Services Group
San Diego Supercomputer Center
University of California, San Diego

CIML Summer Institute
Tuesday, June 27th, 2023
11:10 AM - 12:30 AM PT

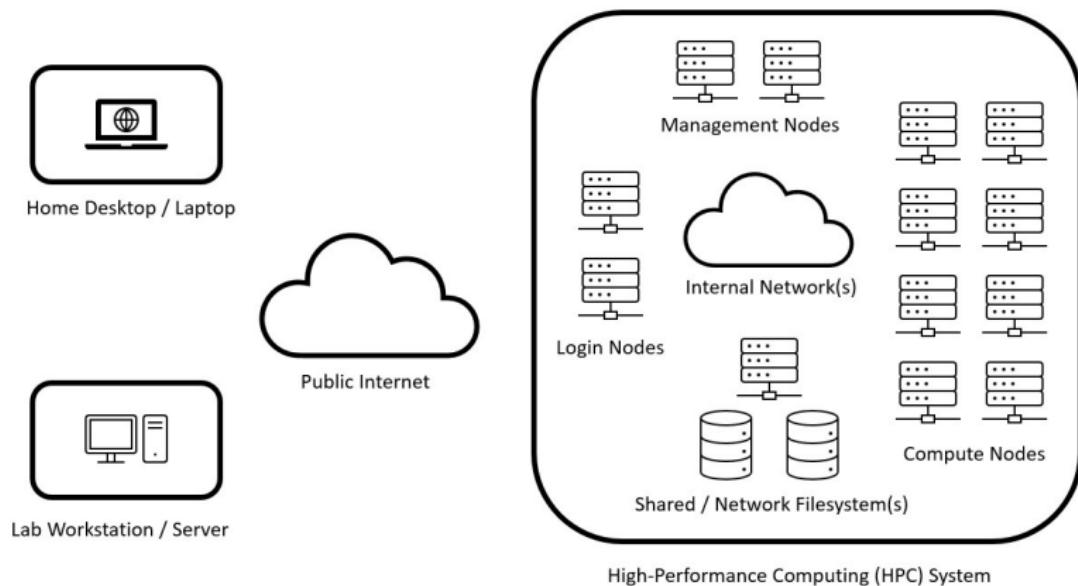
Getting Started with Batch Job Scheduling: Slurm Edition

- ▶ What is a batch job?
- ▶ What is Slurm?
- ▶ How to write and run your first batch job on Expanse

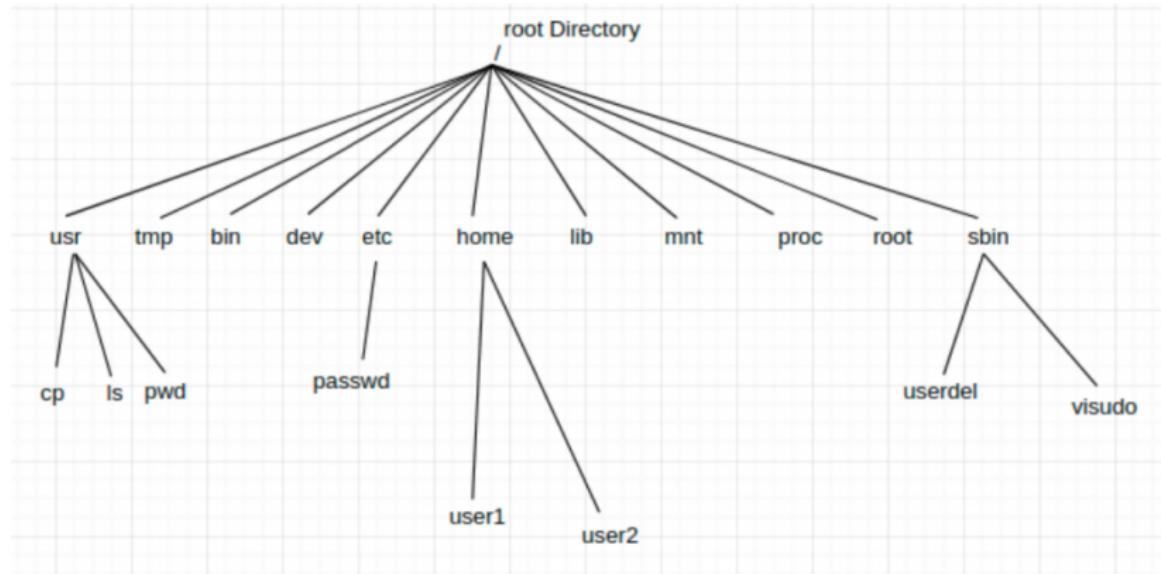


<https://slurm.schedmd.com>

HPC System Architecture: Conceptual



Exercise 1: Where is your HOME?



```
cat /etc/auto.home | grep $USER
```

Shared Network File Systems (NFS)

```
mkanedes@login02:~
```

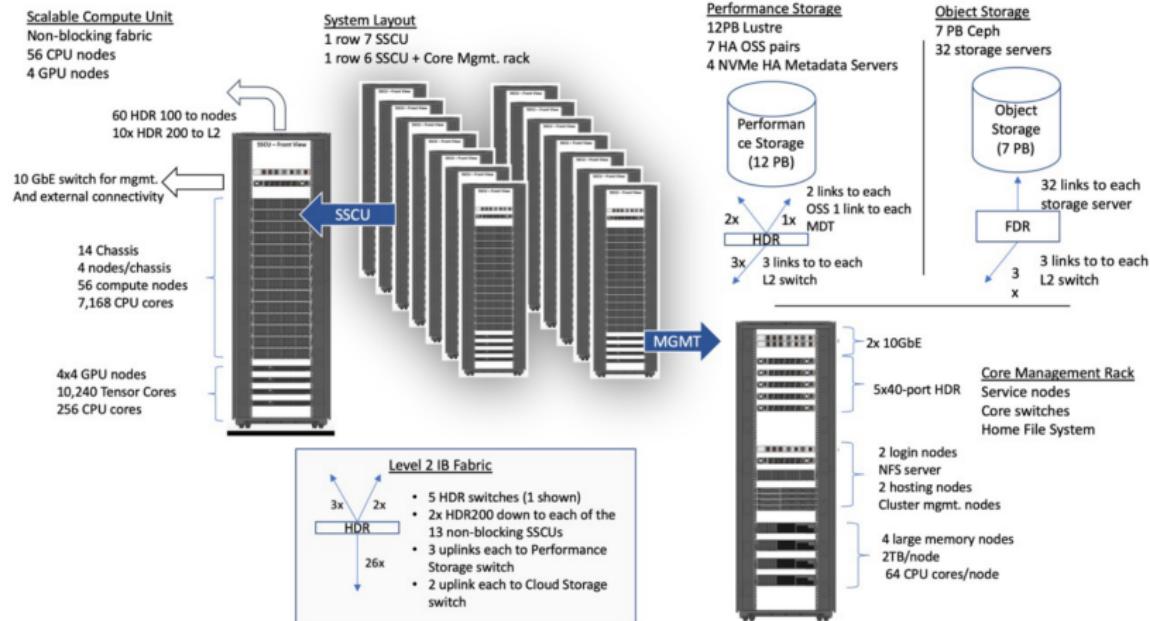


Use the following commands to adjust your environment:

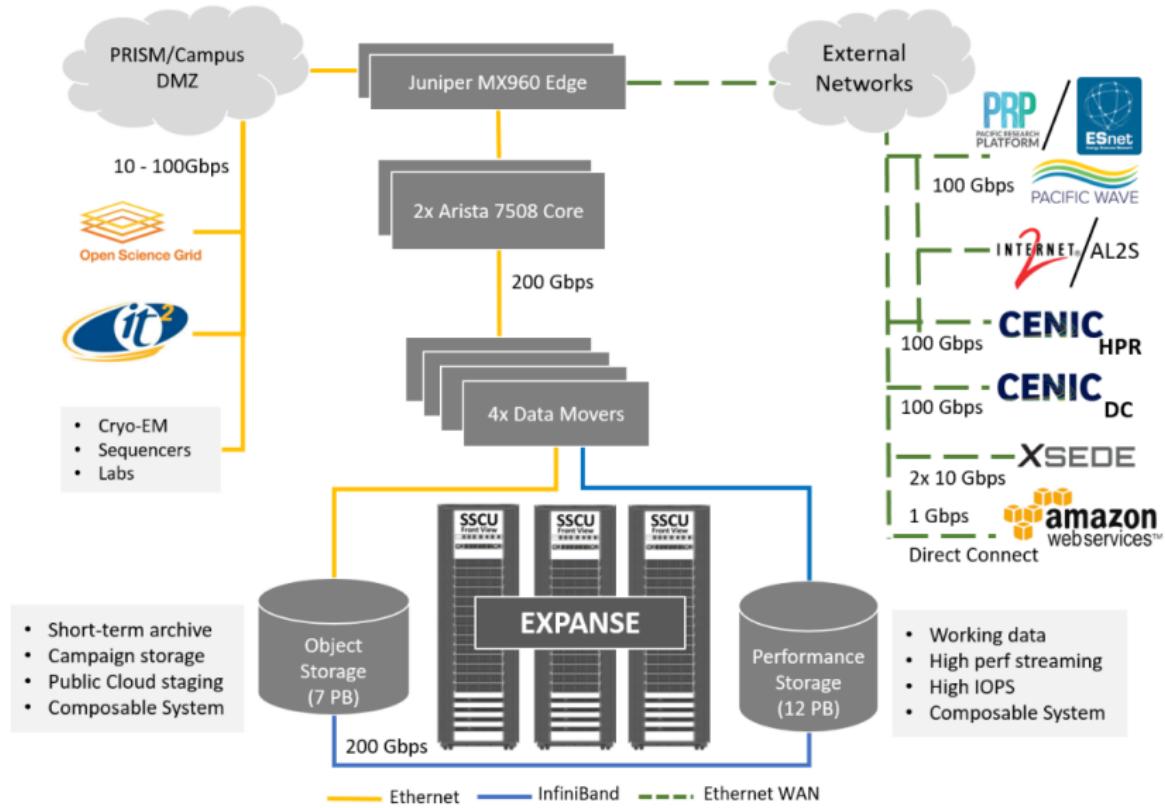
```
'module avail'           - show available modules  
'module add <module>'   - adds a module to your environment for this session  
'module initadd <module>' - configure module to be loaded at every login
```

```
Last login: Sun Jun 25 15:02:41 2023 from 70.166.85.195  
[mkandes@login02 ~]$ cat /etc/auto.home | grep $USER  
mkandes          -fstype=bind :/expanse/nfs/home2/mkandes  
mkandes_test      -fstype=bind :/expanse/nfs/home4/mkandes_test  
[mkandes@login02 ~]$ echo $HOME  
/home/mkandes  
[mkandes@login02 ~]$ pwd  
/home/mkandes  
[mkandes@login02 ~]$ █
```

HPC System Architecture: Not Conceptual



HPC System Architecture: Not Conceptual

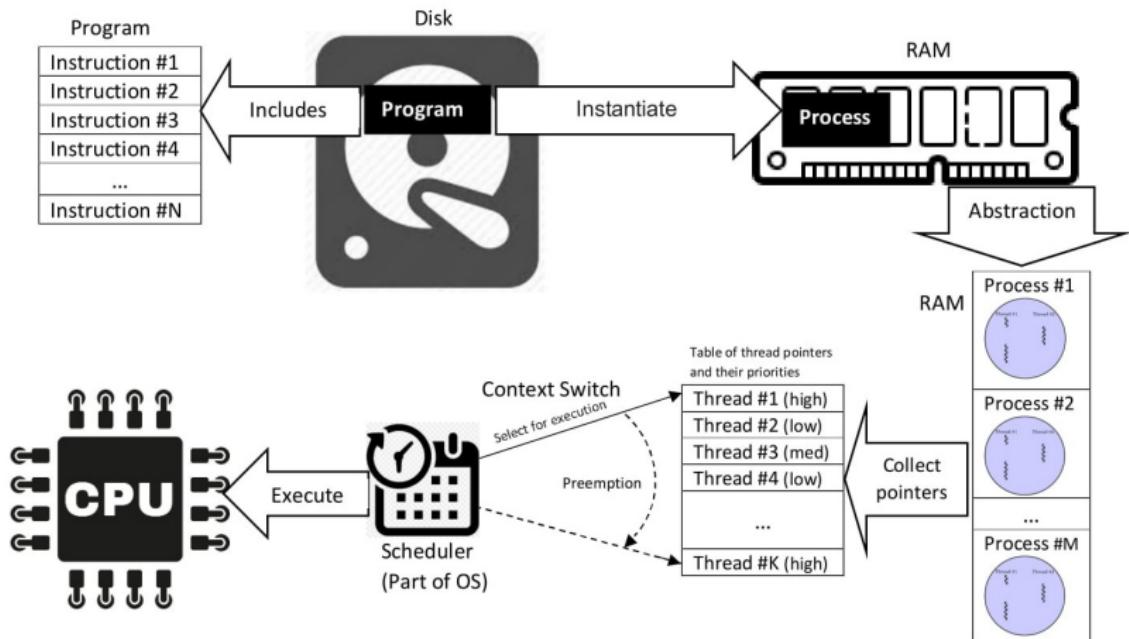


Interactive Computing vs. Batch Processing

Interactive computing is the use of programs that accept input from the user as they execute. Common examples include: word processors, spreadsheet software, web browsers, and social media applications on your smart phone. However, interactive computing and software applications are also used in HPC, mainly for code development and testing as well as real-time data exploration, analysis, and visualization.

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

Scheduling



Scheduling is the action of assigning resources to perform tasks.

Exercise 2: topping the list



top

```
mkandes@login02:~
```

TOP(1) User Commands TOP(1)

NAME
top - display Linux processes

SYNOPSIS
`top [-hv|-bcEHHiOSs1] [-d secs] [-n max] [-u|U user] [-p pid] [-o fld] [-w [cols]]`

The traditional switches `-' and whitespace are optional.

DESCRIPTION
The **top** program provides a dynamic real-time view of a running system. It can display **system** summary information as well as a list of **processes** or **threads** currently being managed by the Linux kernel. The types of system summary information shown and the types, order and size of information displayed for processes are all user configurable and that configuration can be made persistent across restarts.

The program provides a limited interactive interface for process manipulation as well as a much more extensive interface for personal configuration -- encompassing every aspect of its operation.

Manual page top(1) line 1 (press h for help or q to quit)

htop

```
mkanes@login01:~
```

HTOP(1) User Commands HTOP(1)

NAME
htop, pcp-htop - interactive process viewer

SYNOPSIS
`htop [-dCFhpustvH]
pcp htop [-dCFhpustvH] [--host/-h host]`

DESCRIPTION
`htop` is a cross-platform ncurses-based process viewer.

It is similar to `top`, but allows you to scroll vertically and horizontally, and interact using a pointing device (mouse). You can observe all processes running on the system, along with their command line arguments, as well as view them in a tree format, select multiple processes and act on them all at once.

Tasks related to processes (killing, renicing) can be done without entering their PIDs.

`pcp-htop` is a version of `htop` built using the Performance Co-Pilot (PCP) Metrics API (see `PCPIntro(1)`, `PMAPI(3)`), allowing to extend `htop`

Manual page `htop(1)` line 1 (press h for help or q to quit)

Rule #0: You are not alone.

```
mkandes@login01:~
```

kaushiks pts/634	2023-06-13 15:32	(169.232.145.185)
jlspzw pts/636	2023-06-15 09:43	(35.7.5.147)
cppl3992 pts/680	2023-06-25 13:17	(169.232.128.133)
kaushiks pts/655	2023-06-20 19:24	(169.232.145.185)
rbiltz pts/667	2023-06-07 13:38	(192.207.234.194)
mhnguyen pts/668	2023-06-23 12:03	(70.95.76.91)
sazimi pts/669	2023-06-25 12:17	(71.105.165.12)
gcommunityuser pts/673	2023-06-20 13:16	(128.113.130.88)
jaschwa pts/674	2023-06-25 13:07	(208.103.131.104)
jonlyon pts/658	2023-06-25 12:11	(216.249.144.55)
jcannon pts/659	2023-06-06 09:20	(161.130.30.222)
mkandes pts/670	2023-06-25 13:18	(70.166.85.195)
sakhlaghi pts/677	2023-06-20 08:10	(130.74.61.70)
workmanr pts/699	2023-06-21 11:19	(129.109.88.197)
qwinger pts/663	2023-06-21 10:34	(132.239.72.38)
pl201 pts/665	2023-06-25 12:16	(172.59.220.133)
zhai pts/713	2023-06-23 13:41	(95.91.234.58)
wqyvicki pts/709	2023-06-12 11:49	(129.15.66.236)
solen pts/755	2023-06-20 10:18	(165.230.225.217)
lcmoore pts/604	2023-06-25 11:00	(132.239.35.136)
behzadm pts/605	2023-06-08 08:15	(192.17.99.133)
wgyang pts/763	2023-06-20 10:21	(171.65.103.224)
tktran pts/785	2023-06-22 15:04	(129.186.252.112)

```
[mkandes@login01 ~]$ who
```

Rule #1: DO NOT RUN ON LOGIN NODES!

```
mkanedes@login01:~
```

```
top - 09:44:10 up 66 days, 23:19, 136 users, load average: 81.97, 65.30, 51.46
Tasks: 4892 total, 50 running, 4798 sleeping, 43 stopped, 1 zombie
%Cpu(s): 74.4 us, 6.1 sy, 0.0 ni, 18.7 id, 0.4 wa, 0.3 hi, 0.1 si, 0.0 st
MiB Mem : 128120.1 total, 11569.7 free, 40683.4 used, 75867.0 buff/cache
MiB Swap: 12288.0 total, 296.5 free, 11991.5 used. 85713.8 avail Mem

          PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
3571831 rshesha+  20   0 2592084  45316  33104 R 99.7  0.0  23:03.78 lmp_mpi
3571823 rshesha+  20   0 2592148  46000  33856 R 99.4  0.0  23:04.63 lmp_mpi
3571818 rshesha+  20   0 2592140  46112  33836 R 98.4  0.0  23:03.48 lmp_mpi
3571822 rshesha+  20   0 2592156  45132  32852 R 97.7  0.0  23:05.19 lmp_mpi
3571841 rshesha+  20   0 2592168  46336  34020 R 97.1  0.0  23:06.74 lmp_mpi
3571826 rshesha+  20   0 2592160  45840  33560 R 96.4  0.0  23:06.35 lmp_mpi
3571799 rshesha+  20   0 2592172  46076  33780 R 95.1  0.0  23:07.09 lmp_mpi
3571812 rshesha+  20   0 2592164  46748  34448 R 94.5  0.0  23:04.93 lmp_mpi
3571807 rshesha+  20   0 2592168  45712  33412 R 94.2  0.0  23:03.78 lmp_mpi
3571805 rshesha+  20   0 2592160  45780  33460 R 93.9  0.0  23:07.19 lmp_mpi
3571825 rshesha+  20   0 2592168  46060  33768 R 93.9  0.0  23:07.21 lmp_mpi
3571839 rshesha+  20   0 2592160  45616  33476 R 93.9  0.0  23:06.85 lmp_mpi
3571859 rshesha+  20   0 2592140  46520  34228 R 93.5  0.0  23:06.29 lmp_mpi
3571800 rshesha+  20   0 2592172  45868  33380 R 93.2  0.0  23:03.26 lmp_mpi
3571838 rshesha+  20   0 2592176  46088  33924 R 93.2  0.0  23:05.54 lmp_mpi
3571808 rshesha+  20   0 2592176  46324  34024 R 92.9  0.0  23:03.78 lmp_mpi
3571833 rshesha+  20   0 2592168  45692  33264 R 92.9  0.0  23:07.00 lmp_mpi
```

See Rule #0

What type of processes are these on the login nodes?

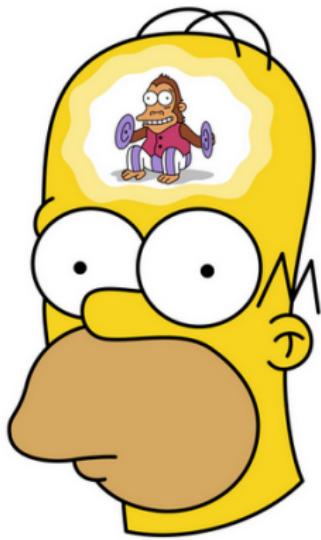


Background vs. Foreground Processes

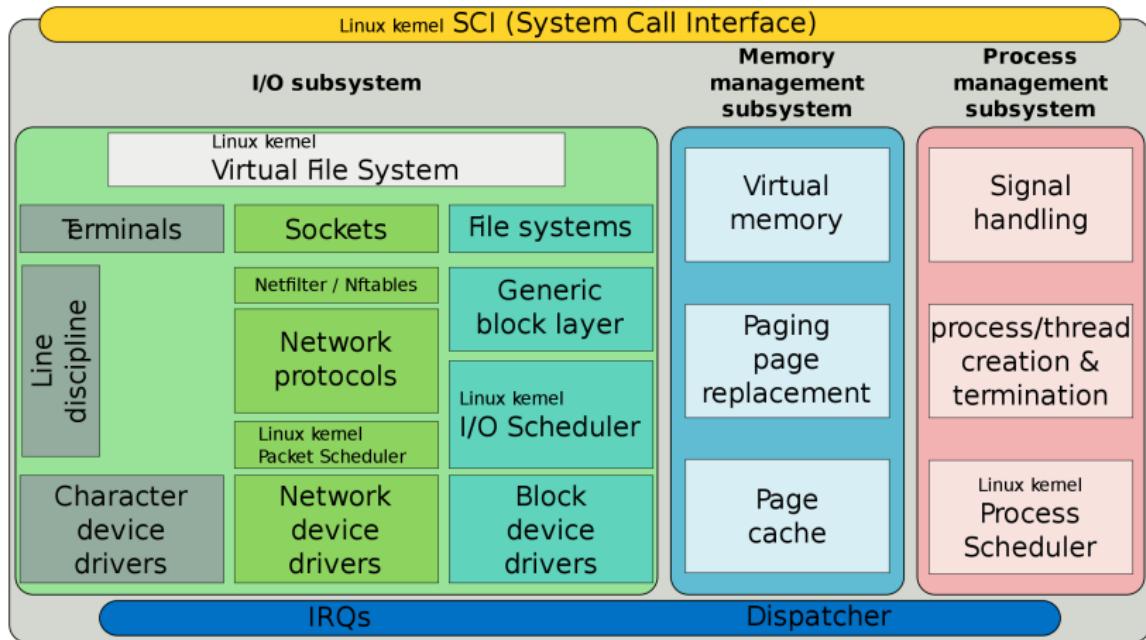
A foreground process is a process that runs a shell command or program immediately upon instantiation by a user and may have its input/output (I/O) directly connected to either a command-line interface (CLI) or a graphical user interface (GUI) to communicate with the user. All shell commands and user programs run as foreground processes by default. Foreground processes may also sometimes be referred to as interactive processes. Examples include your operating system's terminal application, your web browser, and your video conferencing software.

In contrast, a **background process** is a process that runs a program independently of any user interaction. As such, once instantiated, you don't have to wait for the process to complete to execute another one. Background processes may also sometime be referred to as non-interactive processes. Any software application that runs a daemon is utilizing background processes. Examples include the SSH server-side application that provides you remote access to your lab's workstation computer and the web server hosting this tutorial.

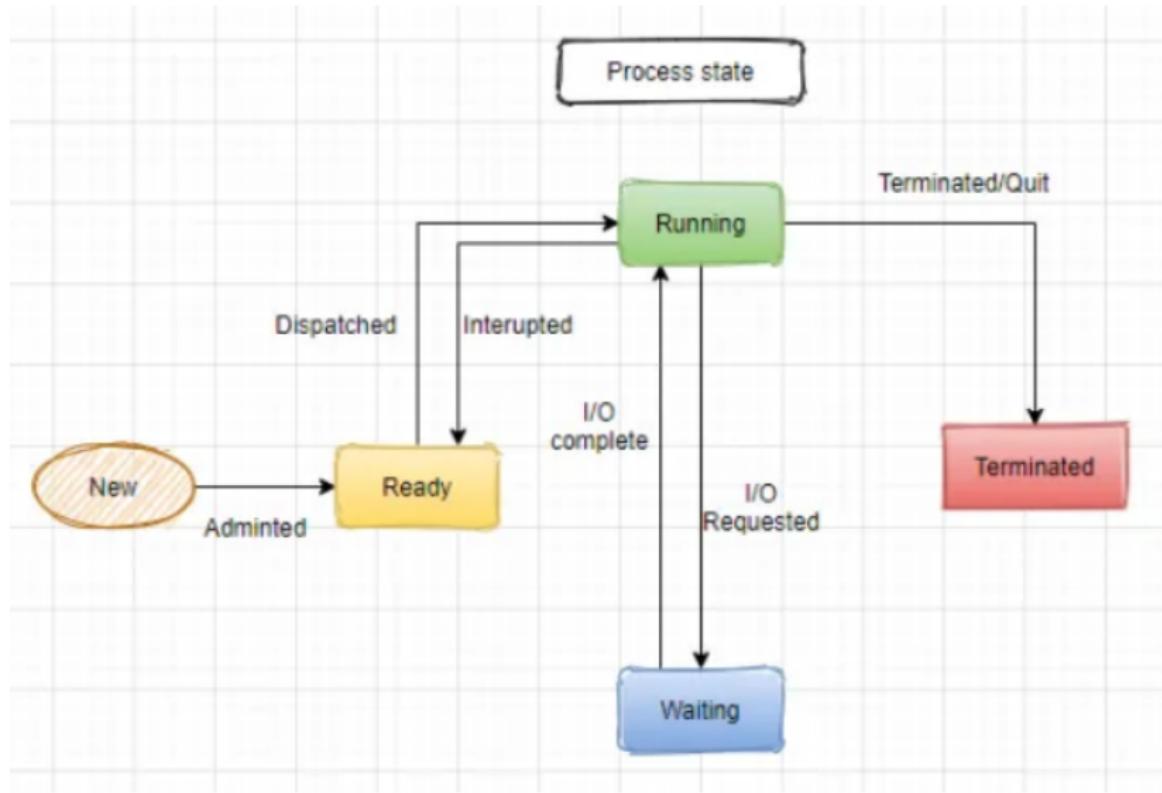
What controls the execution of these processes?



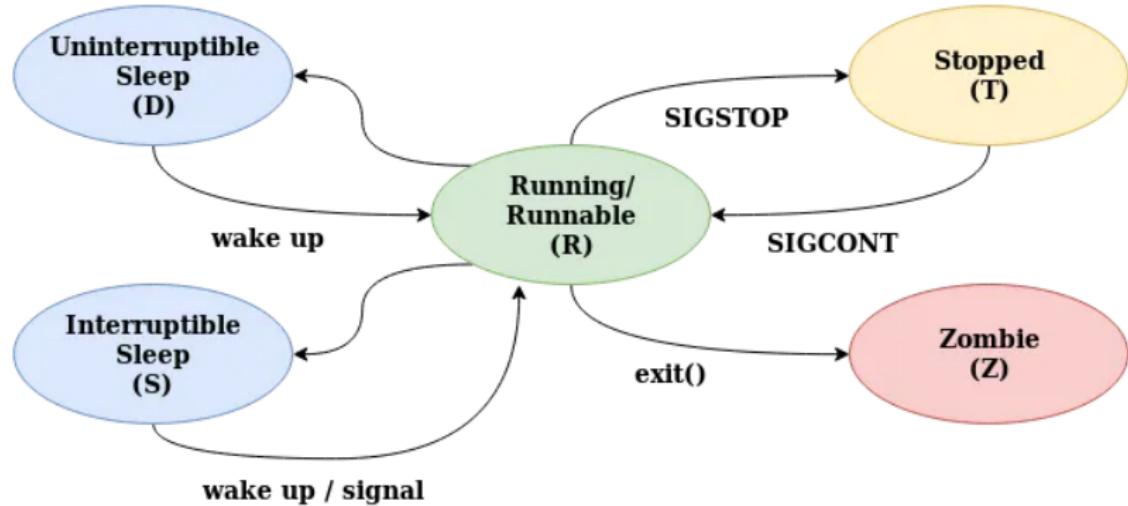
Linux Kernel



Process Life Cycle



top process states



Nohup No Mas: Managing Linux processes



<https://github.com/mkandes/batch-computing>

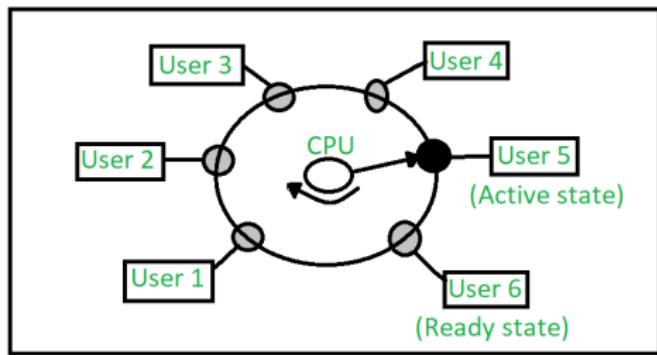
https://education.sdsc.edu/training/interactive/202302_batch_job_scheduling_slurm/

But are these processes being batch processed?

mkandes@login02:~

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2712	dbus	20	0	745464	644076	2680	R	99.7	0.5	16559:37	dbus-da+
1267843	root	26	6	733056	626024	4504	R	99.7	0.5	17553:42	systemd+
62280	root	20	0	8301140	121548	117480	S	92.8	0.1	11641:48	rsyslogd
2384958	root	20	0	472164	249600	25876	S	21.5	0.2	4318:13	systemd+
1	root	26	6	257828	24184	3860	S	12.7	0.0	2932:21	systemd
3790752	mkandes	20	0	71856	9580	4536	R	2.3	0.0	0:02.18	top
3471298	jkleinel	20	0	4876604	106416	976	S	1.6	0.1	176:37.01	python
3789456	xcheng86	20	0	36852	4076	2624	I	1.3	0.0	0:00.70	rsync
63544	root	16	-4	154572	7216	2972	S	0.7	0.0	509:48.41	audited
433401	root	20	0	0	0	0	S	0.7	0.0	248:43.39	kiblnd_+
1013508	aikeliu	20	0	7680748	42548	11732	S	0.7	0.0	133:16.63	stress-+
2453201	aikeliu	20	0	7287232	35808	10664	S	0.7	0.0	121:25.38	stress-+
2814077	nwells	20	0	181472	40692	2688	I	0.7	0.0	0:41.13	rsync
3455962	mmitchell	20	0	5815020	52872	13688	S	0.7	0.0	65:50.26	fermion+
3799989	root	20	0	0	0	0	I	0.7	0.0	0:00.07	ll_sa_3+
428720	dburns	20	0	142436	1736	1156	D	0.3	0.0	116:47.74	plumed
433402	root	20	0	0	0	0	S	0.3	0.0	248:43.27	kiblnd_+

Linux Process Scheduler - Completely Fair Scheduler (CFS)



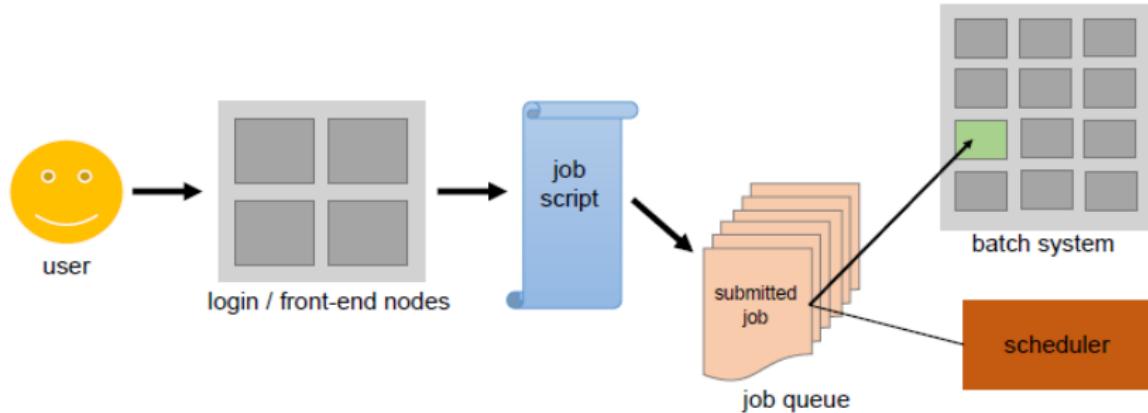
See also: [Time-sharing](#)

CFS aims to maximize overall CPU utilization, while also maximizing interactive performance. The amount of time for a given task on a processor is computed dynamically as the scheduling context changes over the system's lifetime.

Comet - Expanse's Predecessor



What is a batch job scheduler?



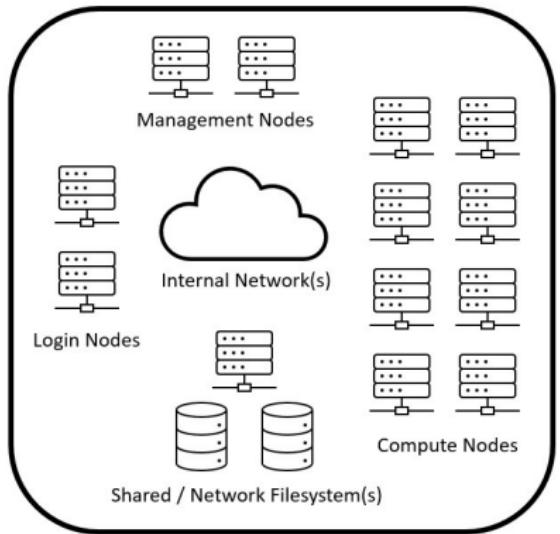
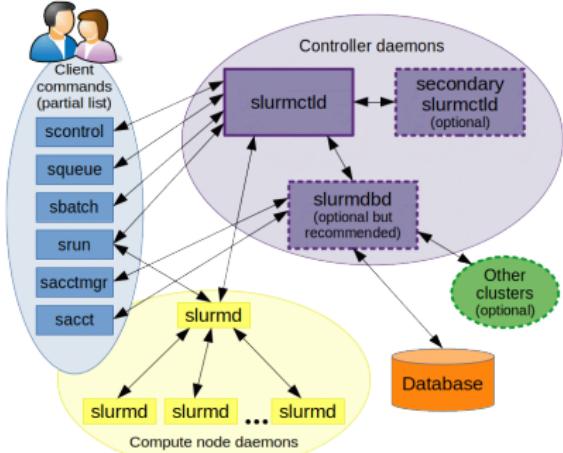
A batch job scheduler is software that controls and tracks where and when batch jobs submitted to a computing system will eventually run on its compute resources.

What is a batch job?

A batch job is a prescribed sets of commands that are executed a process or multiple processes, which may also be threaded, on a certain type or set of dedicated compute resources in a computing system for a given amount of time without user interaction.

What is a batch job script?

Slurm Workload Manager



High-Performance Computing (HPC) System

Exercise 3: Looking squeue and beyond

A screenshot of a Google search results page for the query "askew". The search bar at the top contains the word "askew". Below the search bar, there are several navigation links: Images, News, Left, Spelling, More, Meaning, Synonym, Thrown, In a sentence, All filters, and Tools. The main content area shows approximately 22,900,000 results found in 0.21 seconds. A dictionary definition for "askew" is displayed, defining it as "not in a straight or level position." It includes examples like "Her hat was slightly askew" and "The plan went sadly askew". Below the definition, there are similar words: crooked, lopsided, tilted, angled, oblique, skew, and skewed. A "Feedback" link is located below the definition. At the bottom of the page, there is a "People also ask" section with questions such as "What does the expression askew mean?", "Is it asue or askew?", "What does askew thinking mean?", and "What does thrown askew mean?".

squeue

```
mkanedes@login02:~ squeue(1) Slurm Commands squeue(1)

NAME
  squeue - view information about jobs located in the Slurm scheduling queue.

SYNOPSIS
  squeue [OPTIONS...]

DESCRIPTION
  squeue is used to view job and job step information for jobs managed by Slurm.

OPTIONS
  -A, --account=<account_list>
    Specify the accounts of the jobs to view. Accepts a comma separated list of account names. This has no effect when listing job steps.

  -a, --all
    Display information about jobs and job steps in all partitions. This causes information to be displayed about partitions that are configured as hidden, partitions that are unavailable to a

  Manual page squeue(1) line 1 (press h for help or q to quit)
```

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

sinfo

```
mkandes@login02:~ sinfo(1) Slurm Commands sinfo(1)

NAME
    sinfo - View information about Slurm nodes and partitions.

SYNOPSIS
    sinfo [OPTIONS...]

DESCRIPTION
    sinfo is used to view partition and node information for a system running Slurm.

OPTIONS
    -a, --all
        Display information about all partitions. This causes information to be displayed about partitions that are configured as hidden and partitions that are unavailable to the user's group.

    -M, --clusters=<string>
        Clusters to issue commands to. Multiple cluster names may be comma separated. A value of 'all' will query all clusters. Note that the SlurmDBD must be up for this option to work properly. This option implicitly sets the --local option.

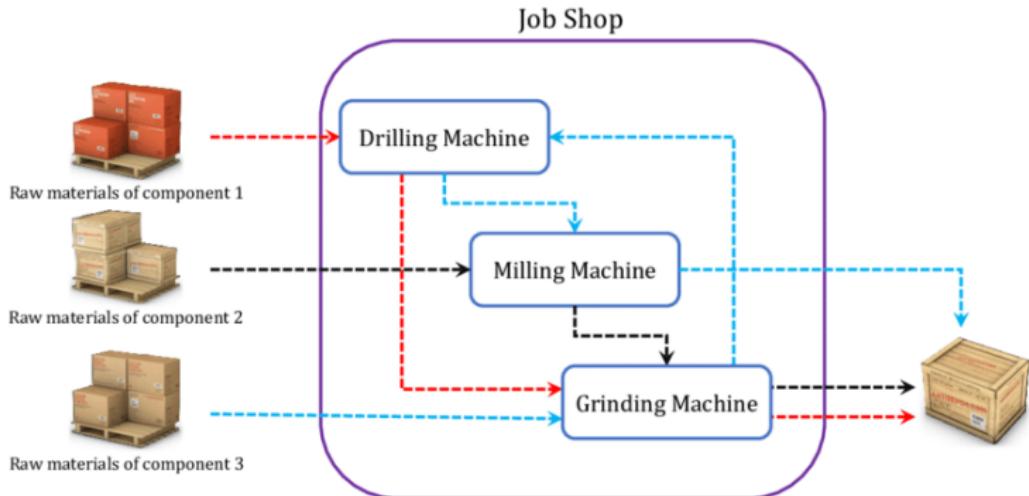
Manual page sinfo(1) line 1 (press h for help or q to quit)
```

Scheduler Rosetta Stone

User Commands	PBS/Torque	Slurm	LSF	SGE	LoadLeveler
Job submission	qsub [script_file]	sbatch [script_file]	bsub [script_file]	qsub [script_file]	lsubmit [script_file]
Job deletion	qdel [job_id]	scancel [job_id]	bkill [job_id]	qdel [job_id]	lcancel [job_id]
Job status (by job)	qstat [job_id]	queue [job_id]	bjobs [job_id]	qstat -u '^' [job_id]	lq-u [username]
Job status (by user)	qstat-u [user_name]	queue-u [user_name]	bjobs-u [user_name]	qstat -u [user_name]	lq-u [user_name]
Job hold	qhold [job_id]	scontrol hold [job_id]	bstop [job_id]	qhold [job_id]	lhold-r [job_id]
Job release	qril [job_id]	scontrol release [job_id]	bresume [job_id]	qril [job_id]	lhold-r [job_id]
Queue list	qstat -Q	queue	bqueues	qconf-sq	lclass
Node list	pnodes -l	info -N OR scontrol show nodes	bhosts	qhost	lstatus -L machine
Cluster status	qstat -a	info -N	bqueues	qhost -q	lstatus -L cluster
GUI	xpbsmon	sview	xifx OR xbatch	qmmon	xload
Environment	PBS/Torque	Slurm	LSF	SGE	LoadLeveler
Job ID	\$PBS_JOBID	\$\$SLURM_JOBID	\$LSLRM_JOBID	\$JOB_ID	\$LOAD_STEP_ID
Submit Directory	\$PBS_O_WORKDIR	\$\$SLURM_SUBMIT_DIR	\$LSLRM_SUBCD	\$SGE_O_WORKDIR	\$LOADL_STEP_INITDIR
Submit Host	\$PBS_O_HOST	\$\$SLURM_SUBMIT_HOST	\$LSLRM_HOST	\$SGE_O_HOST	
Node List	\$PBS_NODEFILE	\$\$SLURM_JOB_NODELIST	\$LSLRM_HOST\$LSB_MCPU_HOST	\$PE_HOSTFILE	\$LOADL_PROCESSOR_LIST
Job Array Index	\$PBS_ARRAYID	\$\$SLURM_ARRAY_TASK_ID	\$LSLRM_JOBINDEX	\$SGE_TASK_ID	
Job Specification	PBS/Torque	Slurm	LSF	SGE	LoadLeveler
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 [count]	node=[count]
CPU Count	-l ppn=[count] OR -l mppwidth=[PE_count]	-n [count]	-n [count]	-pe [PE] [count]	
Wall Clock Limit	-l walltime=[hh:mm:ss]	-l min=[min] OR -l days=hh:mm:ss	-W [hh:mm:ss]	-h rt=[seconds]	wall_clock_limit=[hh:mm:ss]
Standard Output File	-o [file_name]	-o [file_name]	-o [file_name]	-o [file_name]	output=[file_name]
Standard Error File	-e [file_name]	e [file_name]	e [file_name]	-e [file_name]	error=[file_name]
Combine stdio/err	-j oe (both to stdout) OR -j ed (both to stderr)	(use -o without -e)	(use -o without -e)	-j yes	
Copy Environment	-V	-export+[ALL NONE variables]	-V	-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]	configurable default	-r	-r [yes/no]	
Working Directory	N/A	-workingdir=[dir_name]	(submission directory)	-wd [directory]	initialdir=[directory]
Resource Sharing	-l naccesspolicy=singlejob	-exclusive OR -shared	-X	-I exclusive	node_usage=not_shared
Memory Size	-l mem=[MB]	[mem][M G T]	-M [MB]	-mem_free=[memory][K M G]	requirements=(Memory >= [MB])
Account to charge	-W group_list=[account]	-account=[account]	-P [account]	-A [account]	
Tasks Per Node	-l mppnppn [PEs_per_node]	-tasks-per-node=[count]			
CPUs Per Task		-cpus-per-task=[count]			
Job Dependency	-d [job_id]	-depends=[state:job_id]	-w [done exit finish]	-hold_jid [job_id job_name]	
Job Project		-wckey=[name]	-P [name]	-P [name]	
Job host preference		-nodefilter=[nodes] AND/OR --exclude=[nodes]	-m [nodes]	-q [queue]@[node] OR -q [queue]@@[hostgroup]	
Quality Of Service	-l qos=[name]	-qos=[name]			
Job Arrays	-l array_spec	array=[array_spec] (Slurm version 2.6+)	J "name[array_spec]"	-t [array_spec]	
Generic Resources	-l other=[resource_spec]	-gres=[resource_spec]		-l [resource]=[value]	
Licenses	-A "license_spec"	-licenses=[license_spec]	-R "usage[license_spec]"	-l [license]=[count]	
Begin Time	SS*	-begin=YYYY-MM-DD HH:MM:SS*	-b [year]:[month]:[day]:[hour]:[minute]	-a [YYMMDDHHmm]	

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

How does a scheduler work?



See: [Job-Shop Scheduling Problem](#)

How does a scheduler work?



In general, every scheduler has three main goals:

- ▶ Minimize the time between the job submission and finishing the job: no job should stay in the queue for extensive periods of time
- ▶ Optimize CPU utilization: the CPUs of the supercomputer are one of the core resources for a big application; therefore, there should only be few time slots where a CPU is not working
- ▶ Maximize the job throughput: manage as many jobs per time unit as possible

Exercise 4: Hello, My Hostname Is

Hello
my name is

sbatch

```
mkanedes@login02:~
```

sbatch(1) Slurm Commands sbatch(1)

NAME
sbatch - Submit a batch script to Slurm.

SYNOPSIS
sbatch [OPTIONS(0)...] [: [OPTIONS(N)...]] script(0) [args(0)...]

Option(s) define multiple jobs in a co-scheduled heterogeneous job.
For more details about heterogeneous jobs see the document
https://slurm.schedmd.com/heterogeneous_jobs.html

DESCRIPTION
sbatch submits a batch script to Slurm. The batch script may be given to sbatch through a file name on the command line, or if no file name is specified, sbatch will read in a script from standard input. The batch script may contain options preceded with "#SBATCH" before any executable commands in the script. sbatch will stop processing further #SBATCH directives once the first non-comment non-whitespace line has been reached in the script.

sbatch exits immediately after the script is successfully transferred to the Slurm controller and assigned a Slurm job ID. The batch script

Manual page sbatch(1) line 1 (press h for help or q to quit)

srun

```
mkandes@login02:~ srun(1) Slurm Commands srun(1)

NAME
    srun - Run parallel jobs

SYNOPSIS
    srun [OPTIONS(0)... [executable(0) [args(0)...]]] [ : [OPTIONS(N)...]]
    executable(N) [args(N)...]

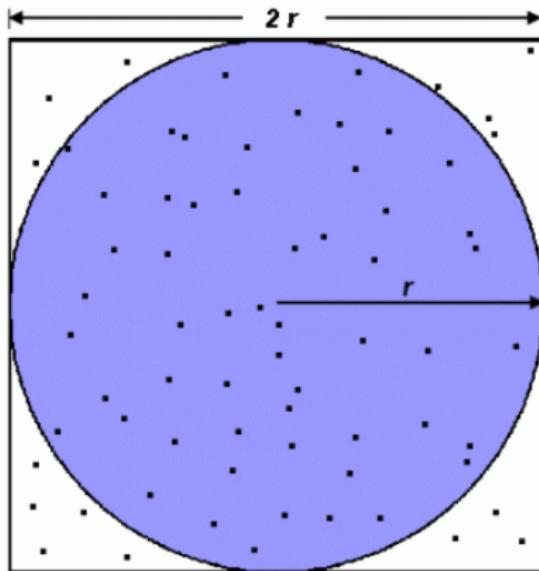
    Option(s) define multiple jobs in a co-scheduled heterogeneous job.
    For more details about heterogeneous jobs see the document
    https://slurm.schedmd.com/heterogeneous\_jobs.html

DESCRIPTION
    Run a parallel job on cluster managed by Slurm. If necessary, srun
    will first create a resource allocation in which to run the parallel
    job.

    The following document describes the influence of various options on
    the allocation of cpus to jobs and tasks.
    https://slurm.schedmd.com/cpu\_management.html

RETURN VALUE
    Manual page srun(1) line 1 (press h for help or q to quit)
```

Exercise 5: Rolling the Dice 4pi



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

$$A_C = \pi r^2$$

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

scancel

```
mkanedes@login02:~ mkandes@login02:~ scancel(1) scancel(1)

scancel(1) Slurm Commands scancel(1)

NAME
    scancel - Used to signal jobs or job steps that are under the control
    of Slurm.

SYNOPSIS
    scancel [OPTIONS...] [job_id[_array_id][._step_id]]
    [job_id[_array_id][._step_id]...]

DESCRIPTION
    scancel is used to signal or cancel jobs, job arrays or job steps. An
    arbitrary number of jobs or job steps may be signaled using job speci-
    fication filters or a space separated list of specific job and/or job
    step IDs. If the job ID of a job array is specified with an array ID
    value then only that job array element will be cancelled. If the job
    ID of a job array is specified without an array ID value then all job
    array elements will be cancelled. While a heterogeneous job is in a
    PENDING state, only the entire job can be cancelled rather than its
    individual components. A request to cancel an individual component of
    a heterogeneous job while in a PENDING state will return an error.
    After the job has begun execution, an individual component can be can-
    celled. A job or job step can only be signaled by the owner of that

Manual page scancel(1) line 1 (press h for help or q to quit)
```

Additional Demos and Q&A ...

Additional References

- ▶ **Slurm Quick Start User Guide:**
<https://slurm.schedmd.com/quickstart.html>
- ▶ **HPCWIKI: Scheduling Basics:**
https://hpc-wiki.info/hpc/Scheduling_Basics
- ▶ **Intro to HPC:**
<https://carpentries-incubator.github.io/hpc-intro>
- ▶ **HPC In a Day:**
<https://psteinb.github.io/hpc-in-a-day>

Questions?

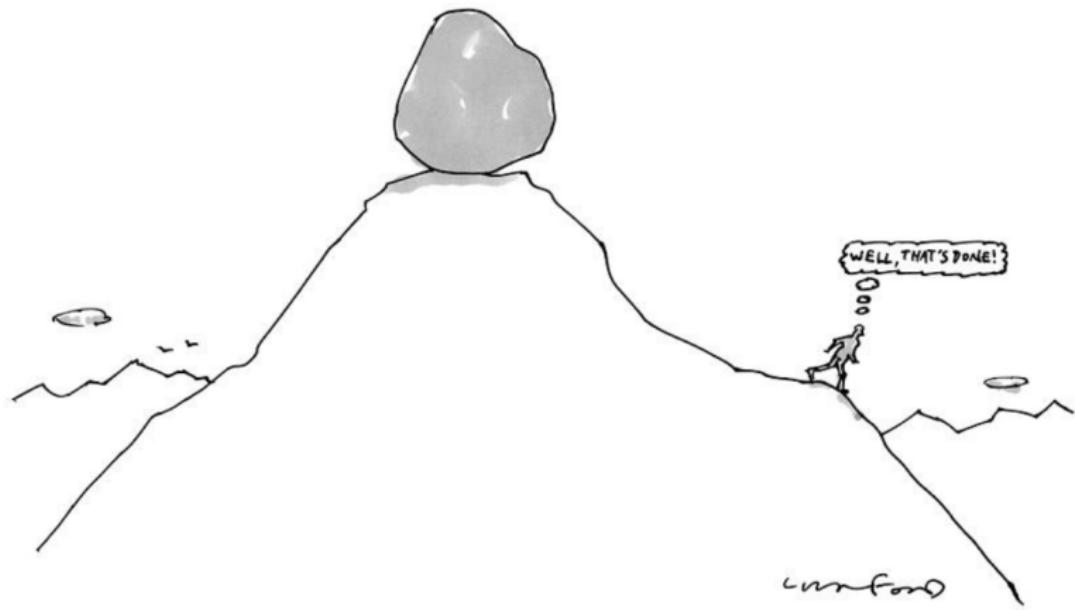


Image Credit: New Yorker - M. Crawford