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

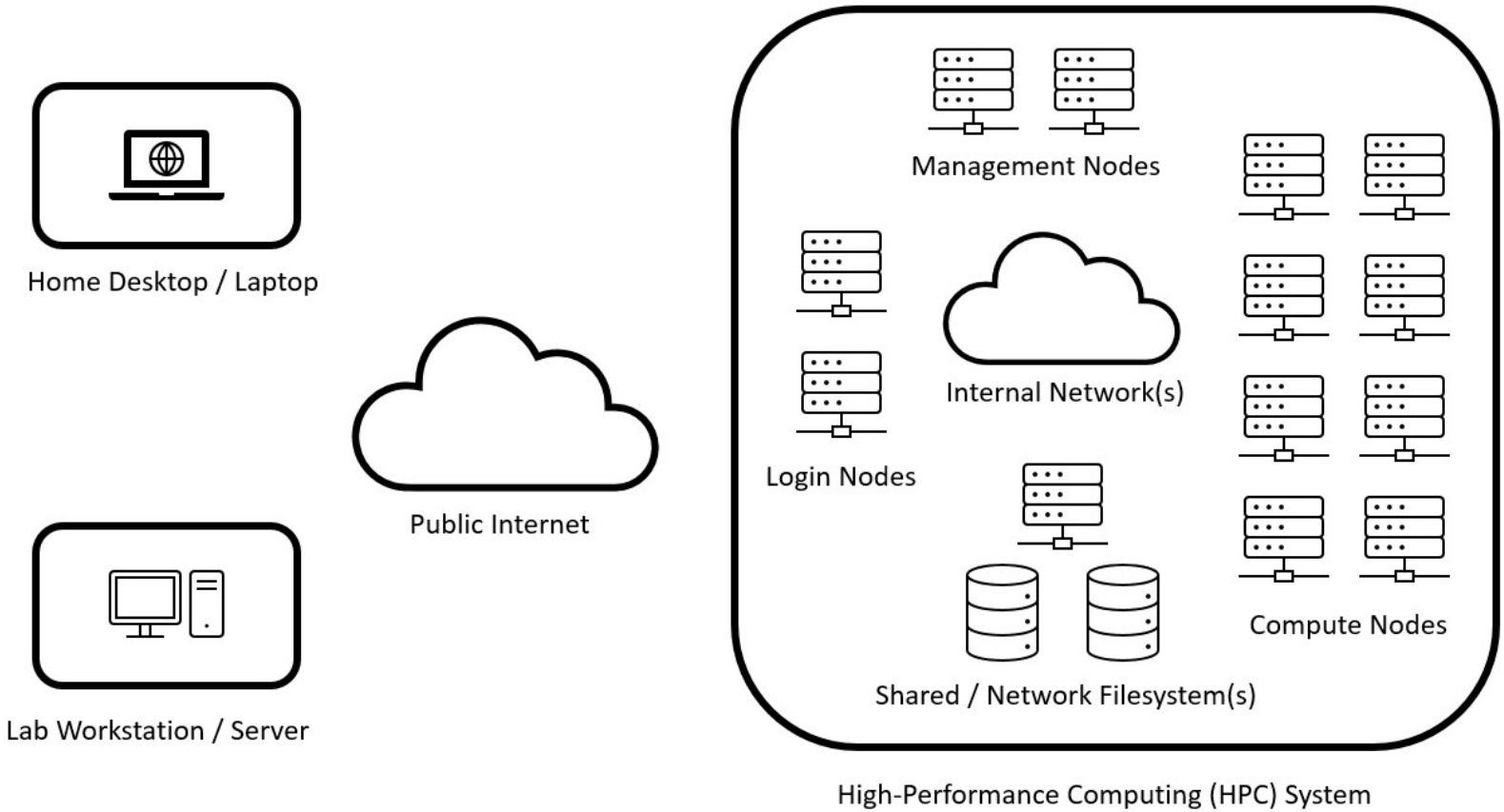
CIML Summer Institute
Tuesday, June 27th, 2023
11:10 AM - 12:30 AM PT

Getting Started with Batch Job Scheduling: Slurm Edition

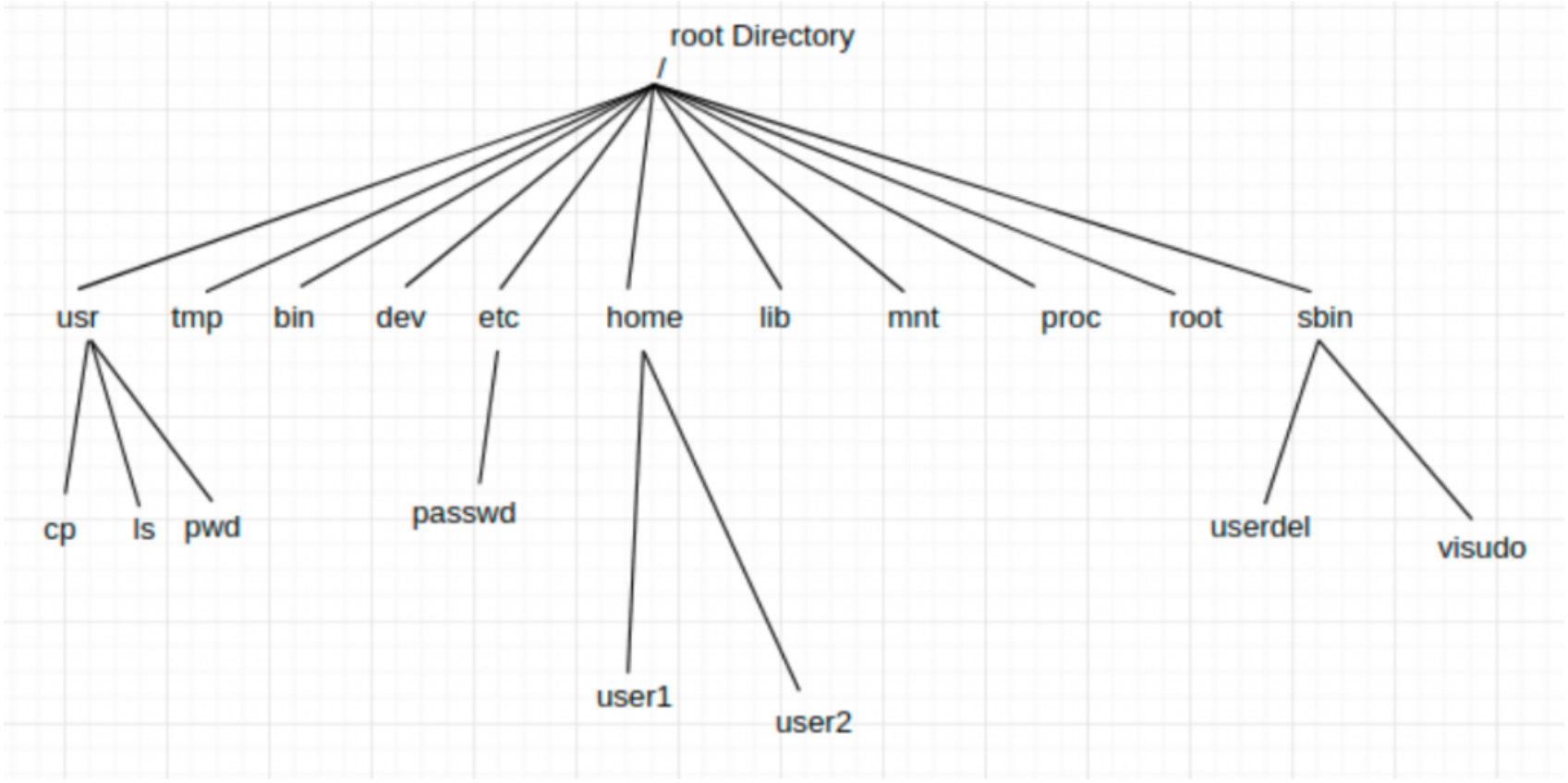
- What is a batch job?
- What is Slurm?
- How to write and run your first batch job on Expanse

https://slurm.schedmd.com

HPC System Architecture: Conceptual

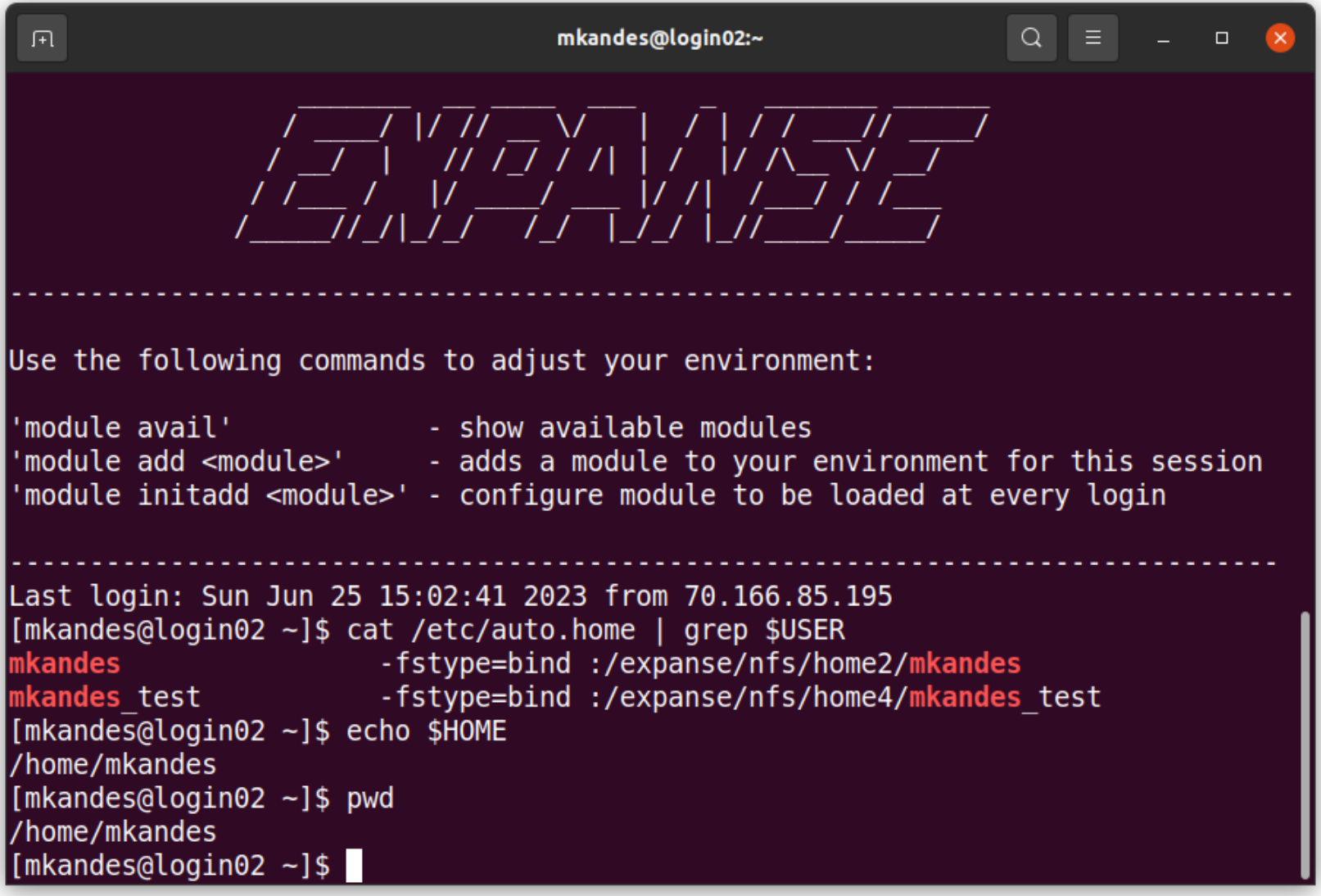


Exercise 1: Where is your HOME?

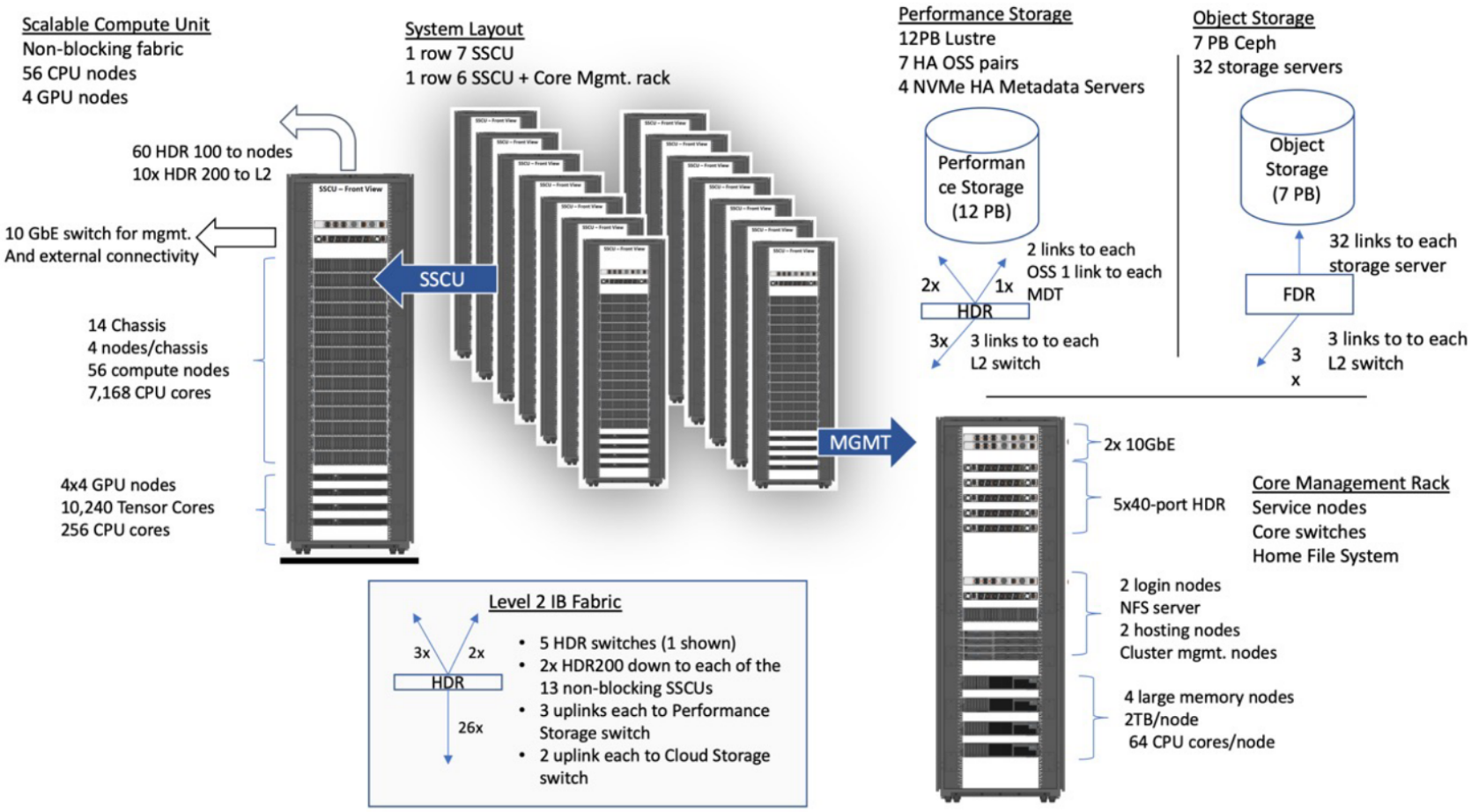



```
cat /etc/auto.home | grep $USER
```

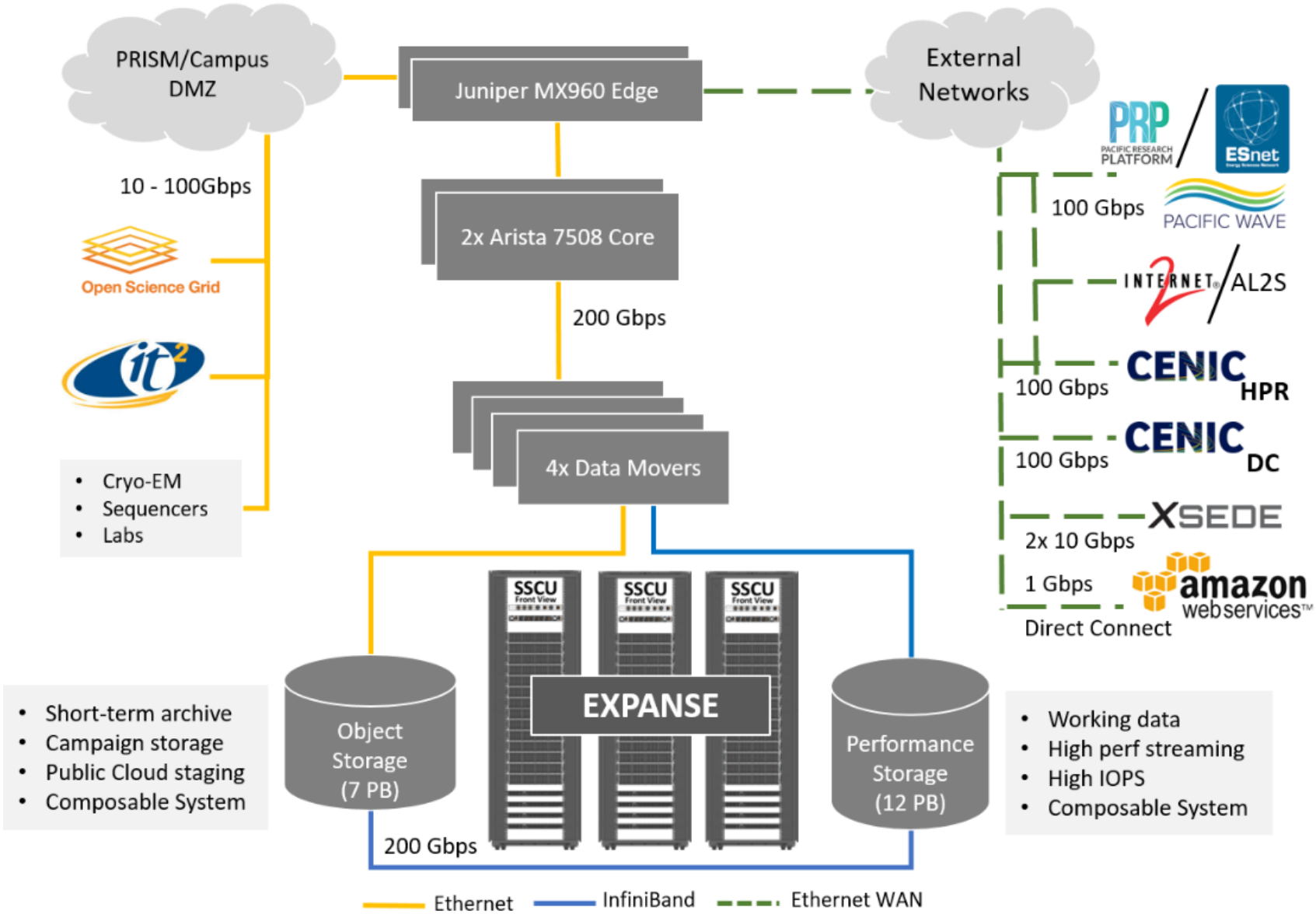
Shared Network File Systems (NFS)



HPC System Architecture: Not Conceptual



HPC System Architecture: Not Conceptual

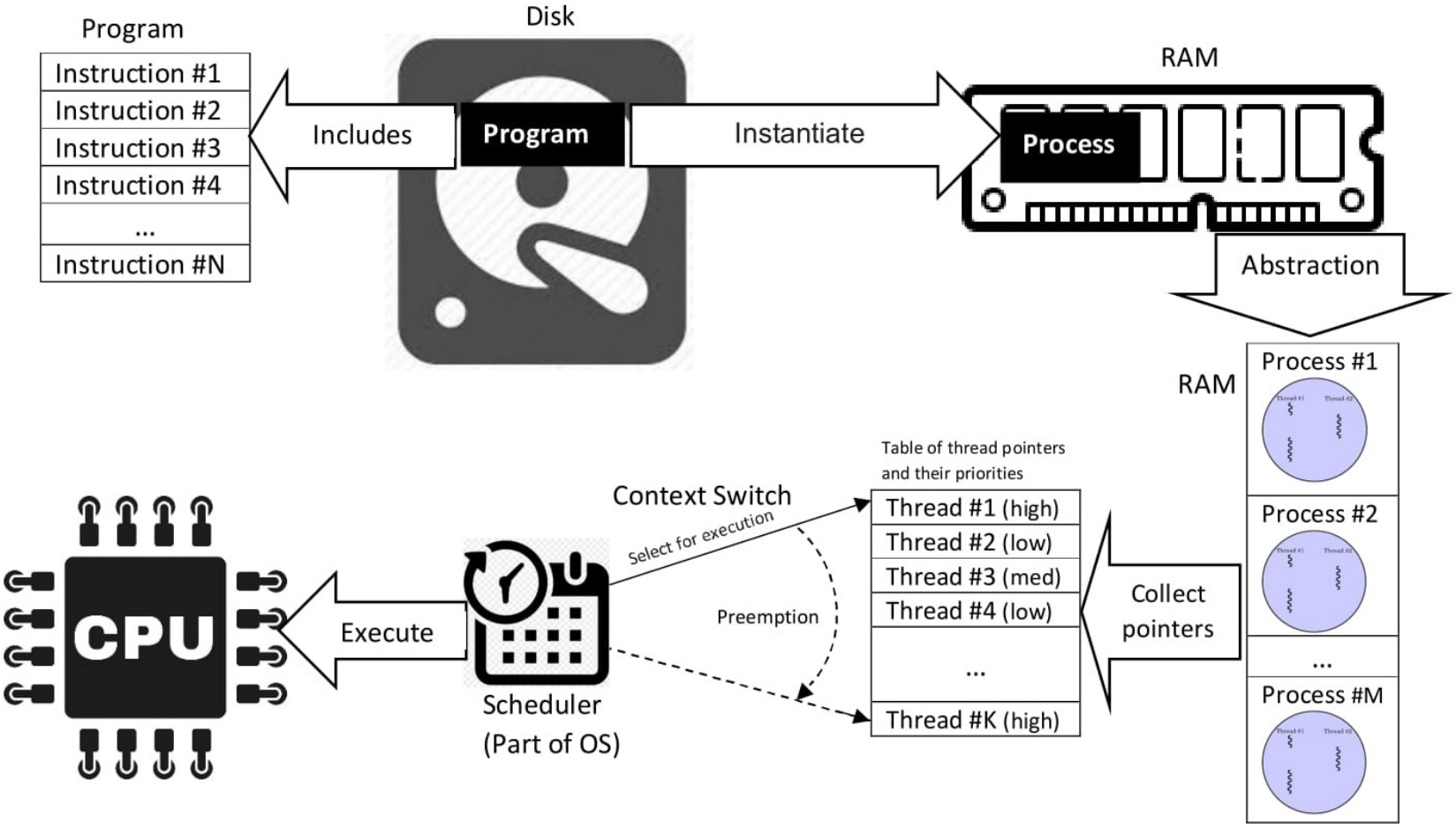


Interactive Computing vs. Batch Processing

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 processesing 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, repetative tasks such as data (re-)processing, but also to allow the shared use of unique compute resources like an HPC system among many users.

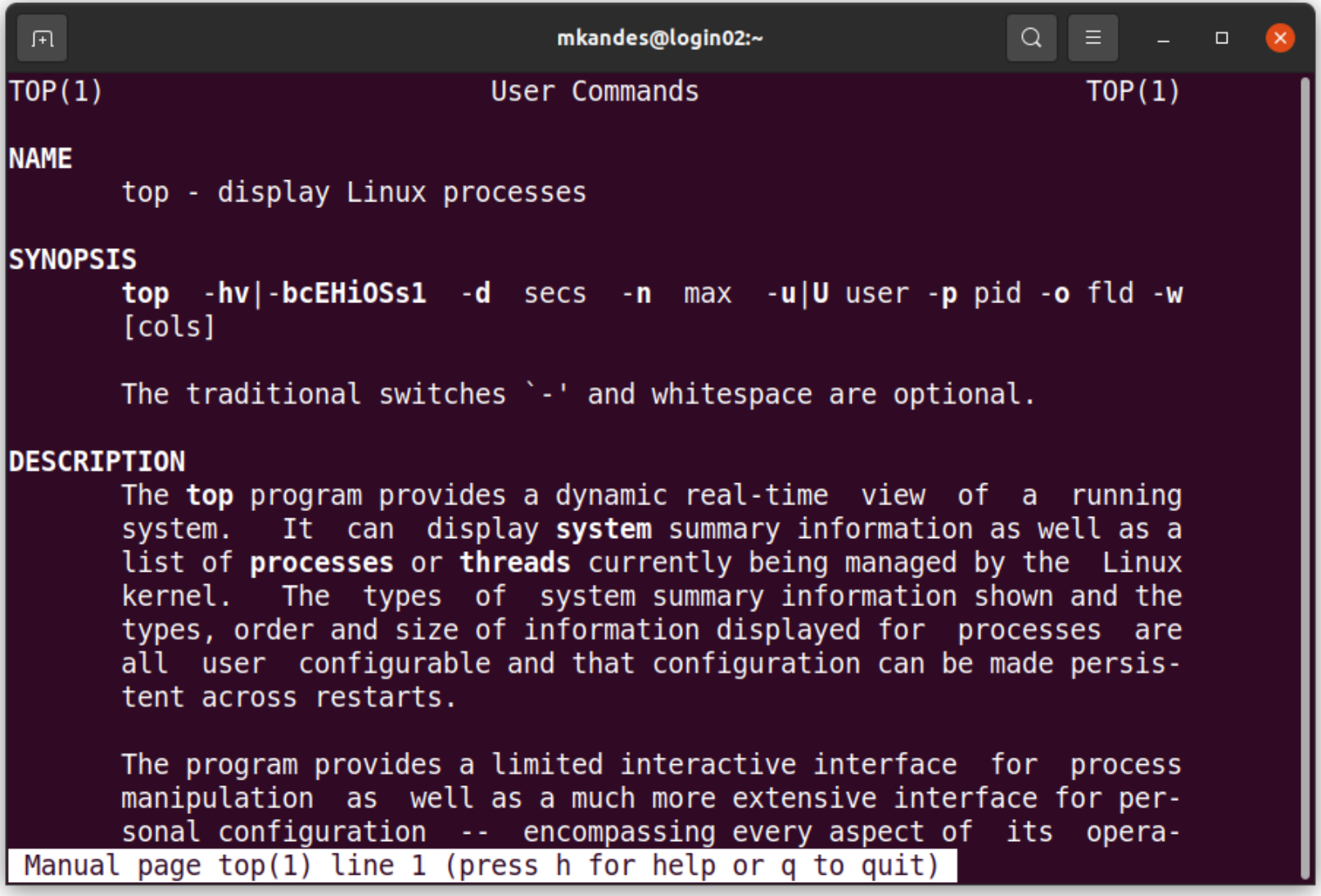
Scheduling



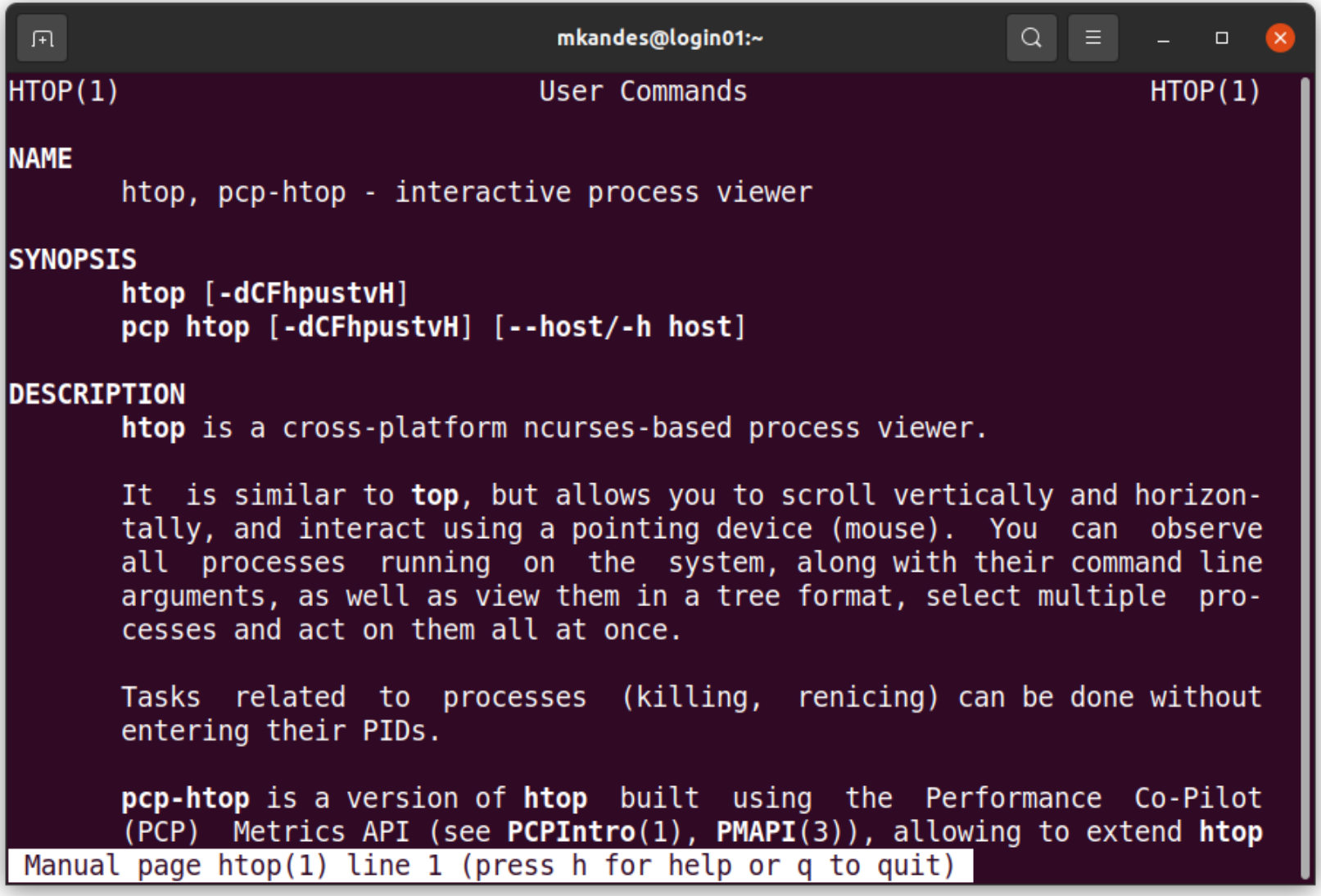
Scheduling is the action of assigning resources to perform tasks.

Exercise 2: topping the list

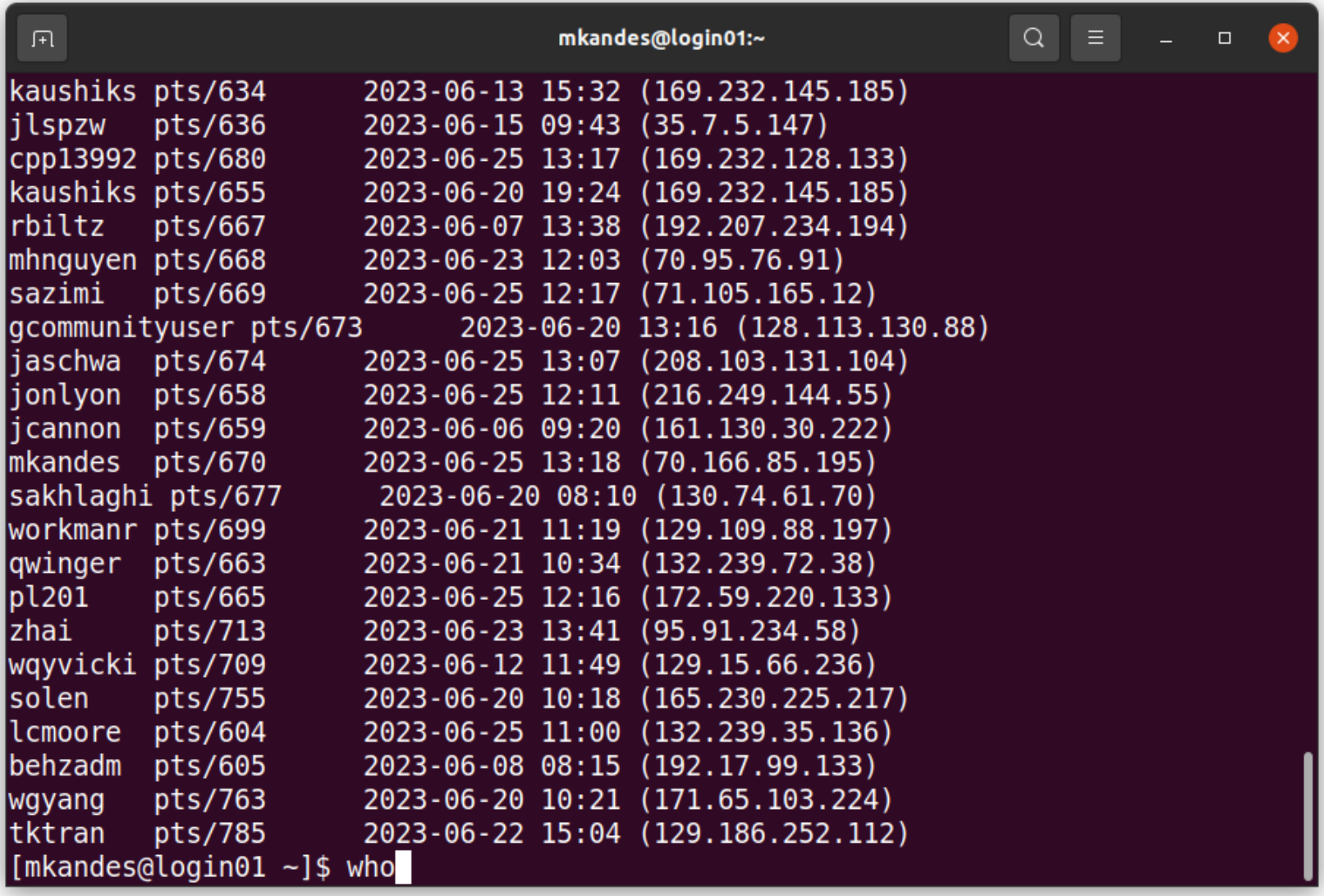
top


mkandes@login02:~
TOP(1) User Commands TOP(1)
NAME
top - display Linux processes
SYNOPSIS
top -hv|-bcEHiOSs1 -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 opera-
Manual page top(1) line 1 (press h for help or q to quit)

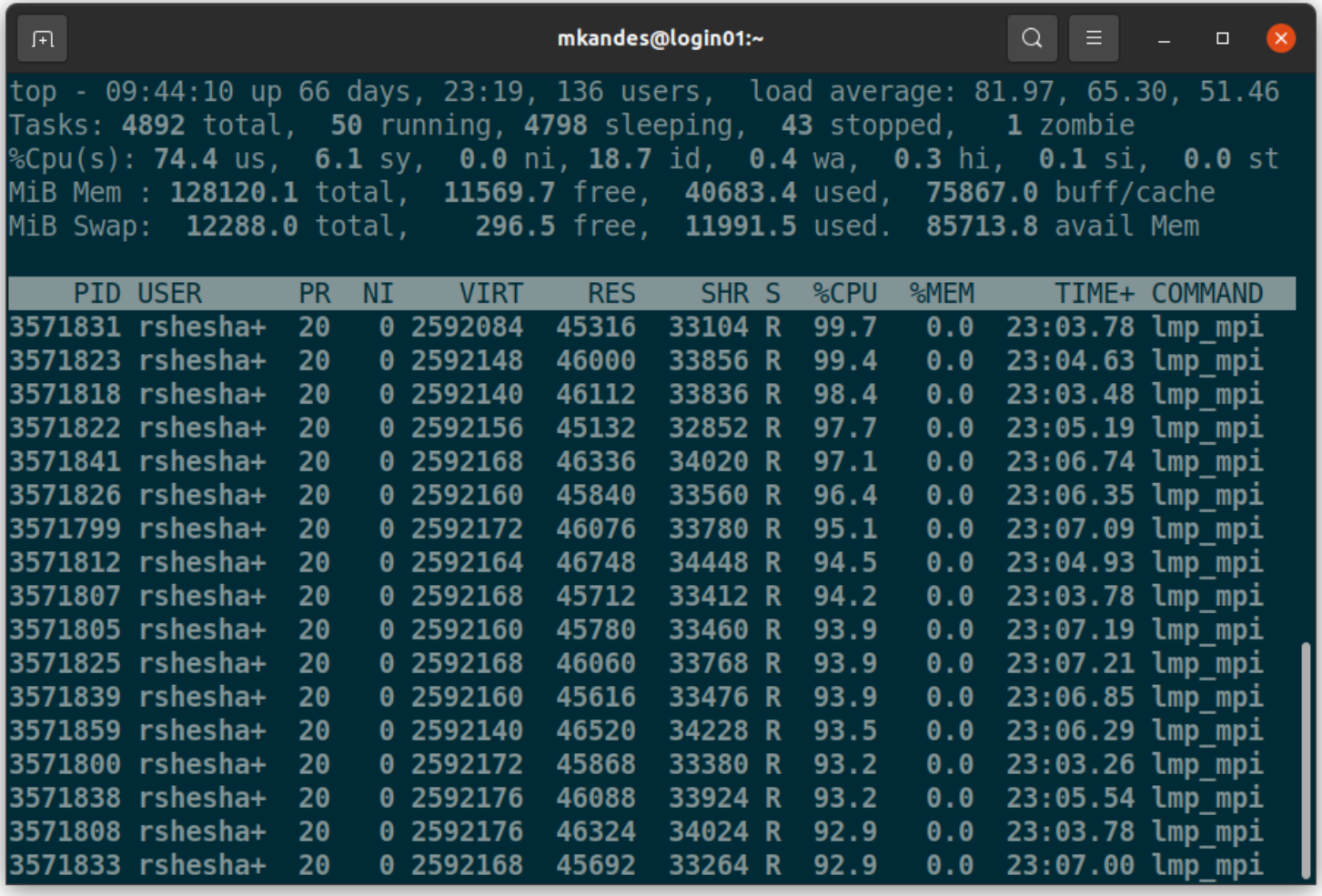
htop


mkandes@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.



Rule #1: DO NOT RUN ON LOGIN NODES!



See Rule #0

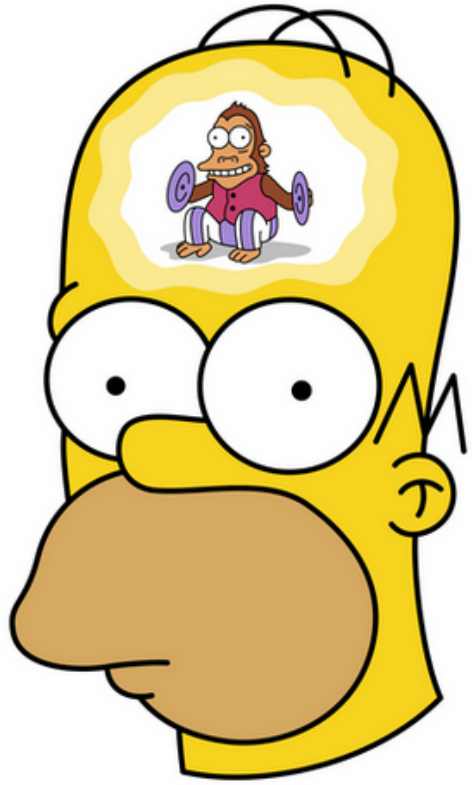
What type of processes are these on the login nodes?

Background vs. Foreground Processes

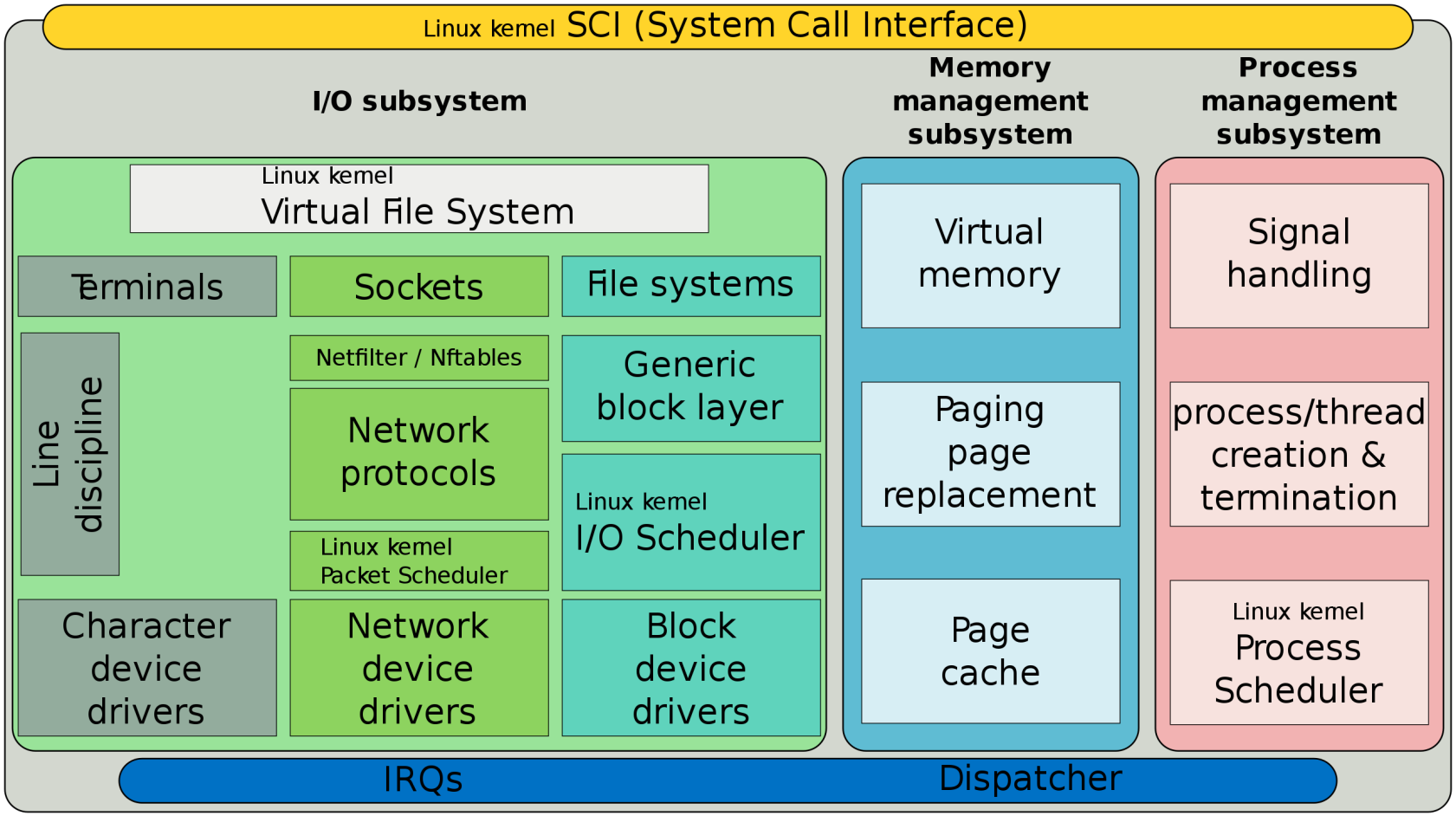
A foreground process is a process that runs a shell command or program immediately upon instantiation by a user and may have its input/output (I/O) directly connected to either a command-line interface (CLI) or a graphical user interface (GUI) to communicate with the user. All shell commands and user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

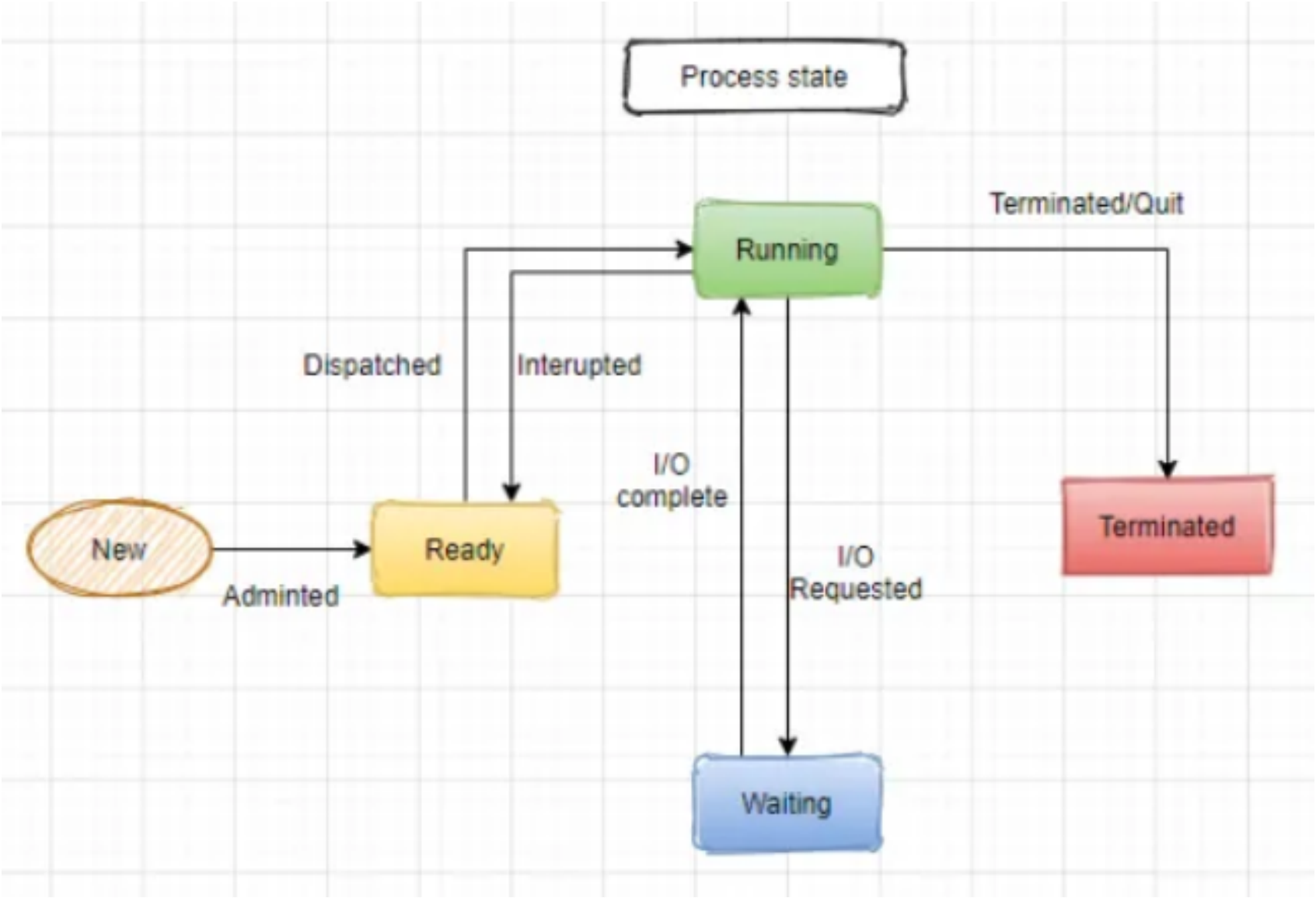
What controls the execution of these processes?



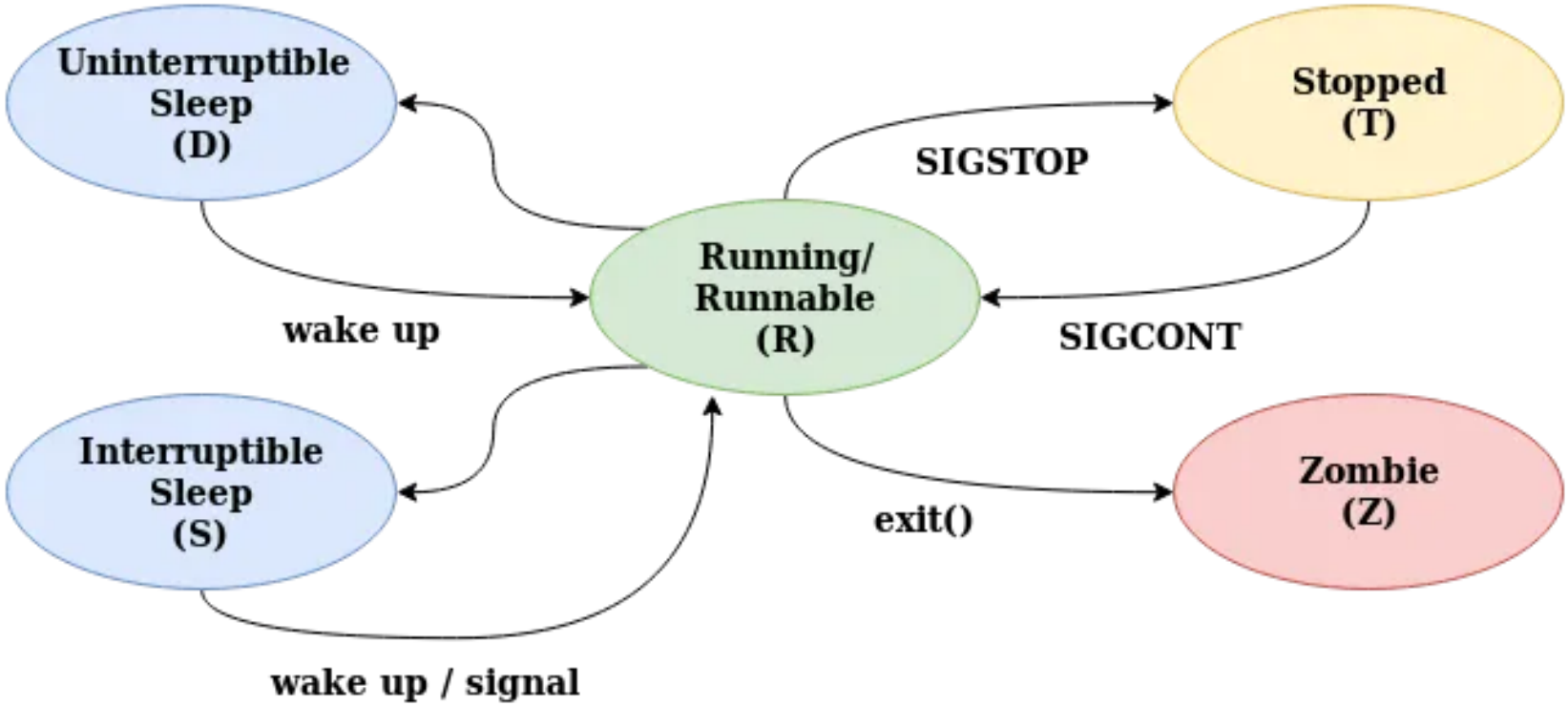
Linux Kernel



Process Life Cycle



top process states

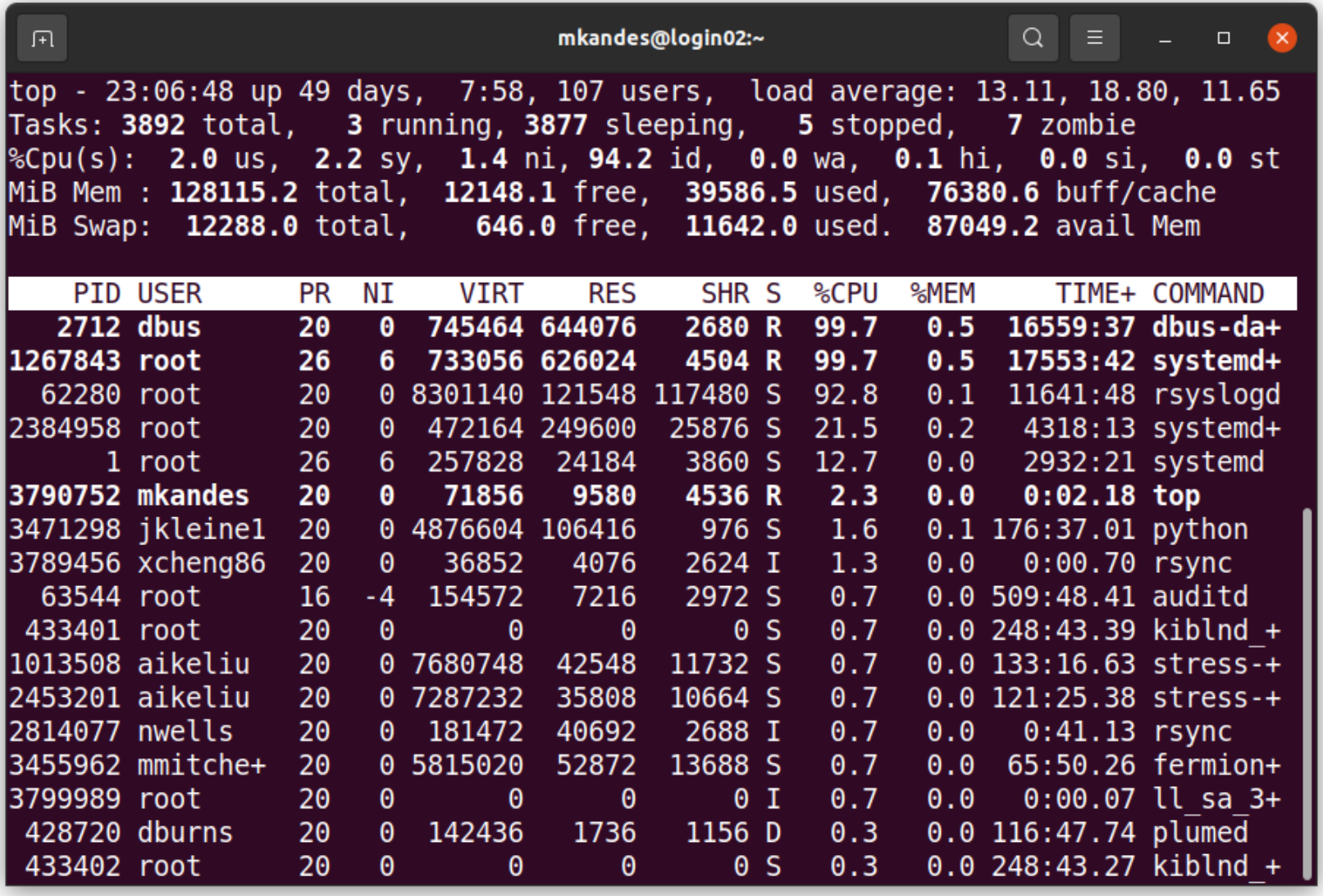


Nohup No Mas: Managing Linux processes

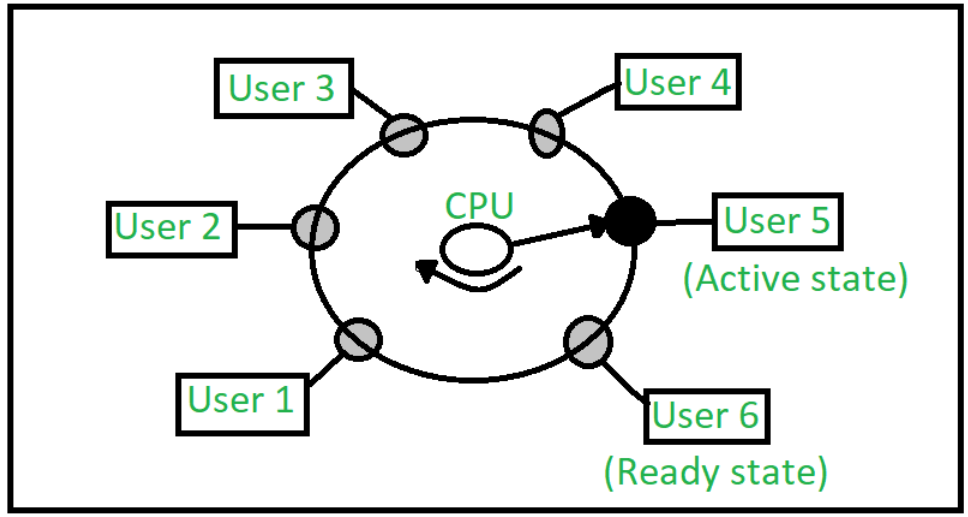
https://github.com/mkandes/batch-computing

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

But are these processes being batch processed?



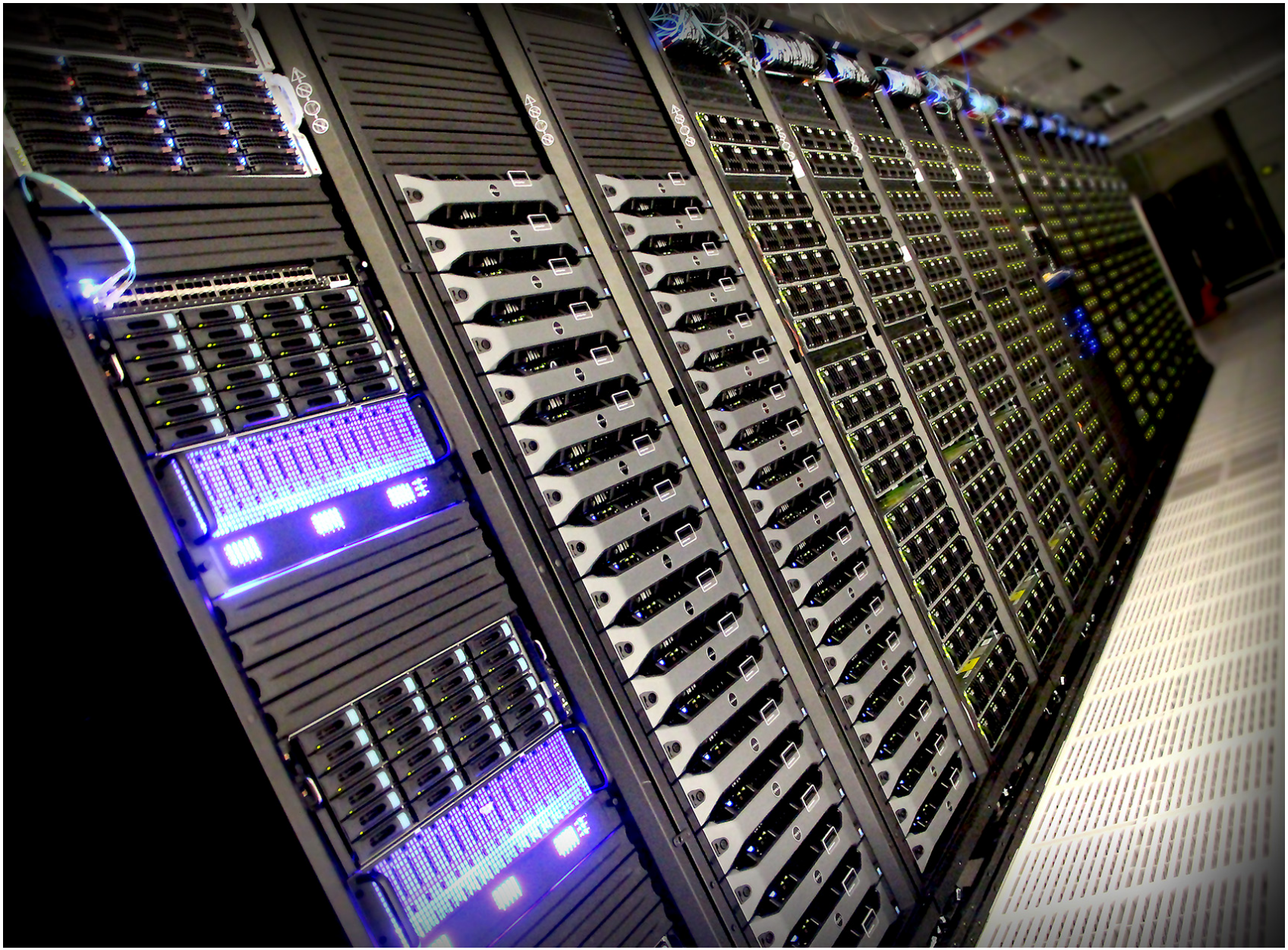
Linux Process Scheduler - Completely Fair Scheduler (CFS)



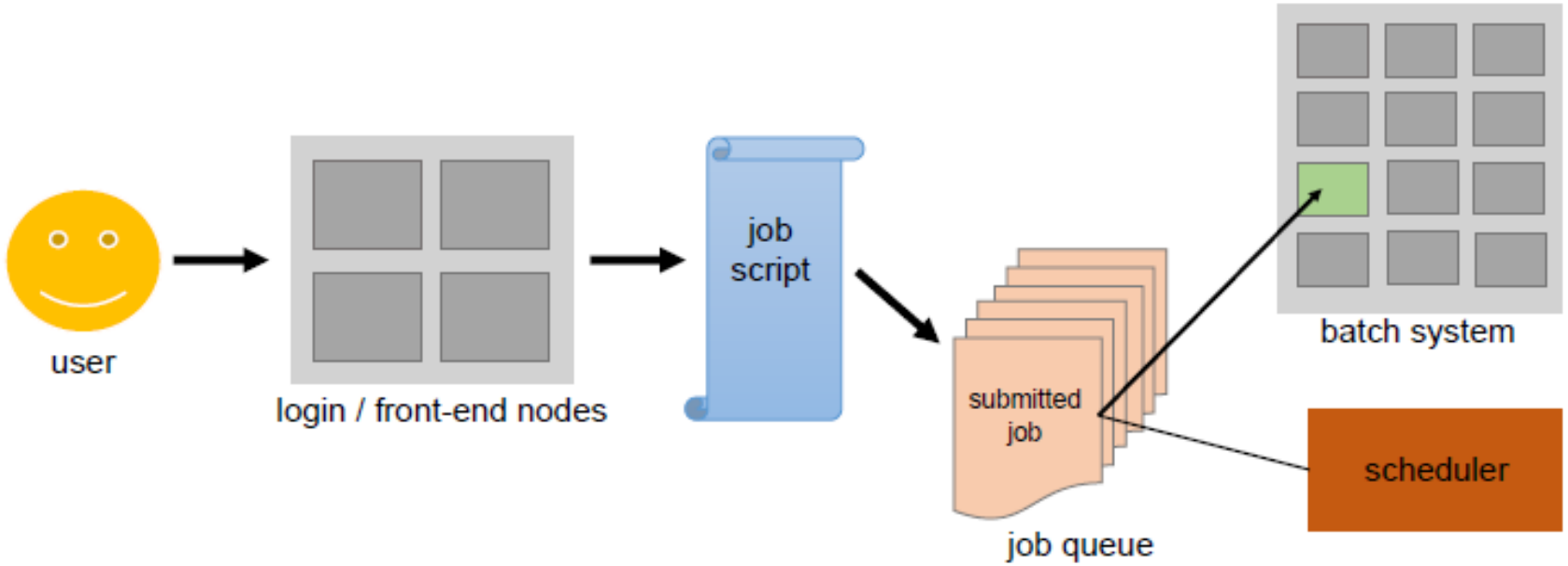
See also: Time-sharing

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?



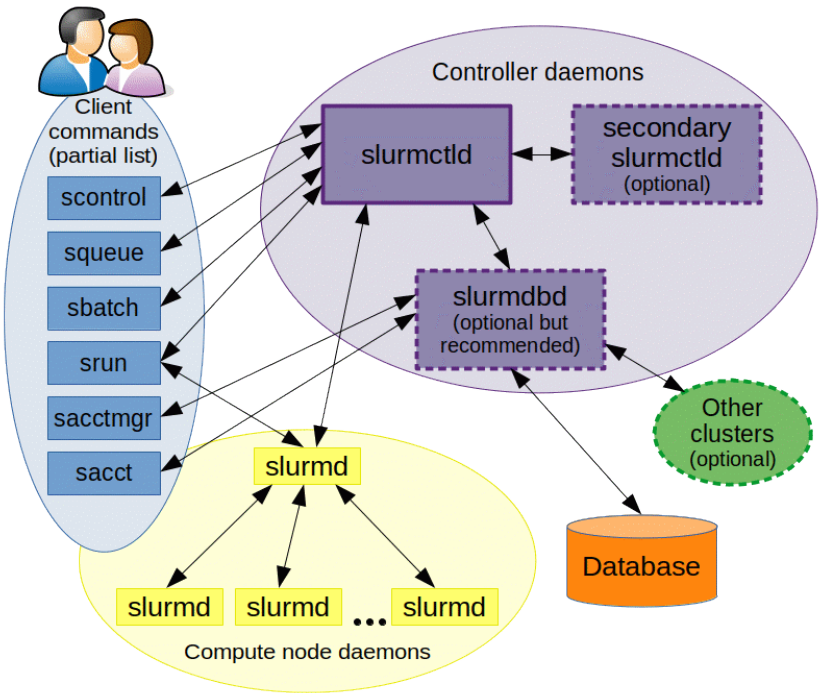
A batch job scheduler is software that controls and tracks where and when batch jobs submitted to a computing system will eventually run on its compute resources.

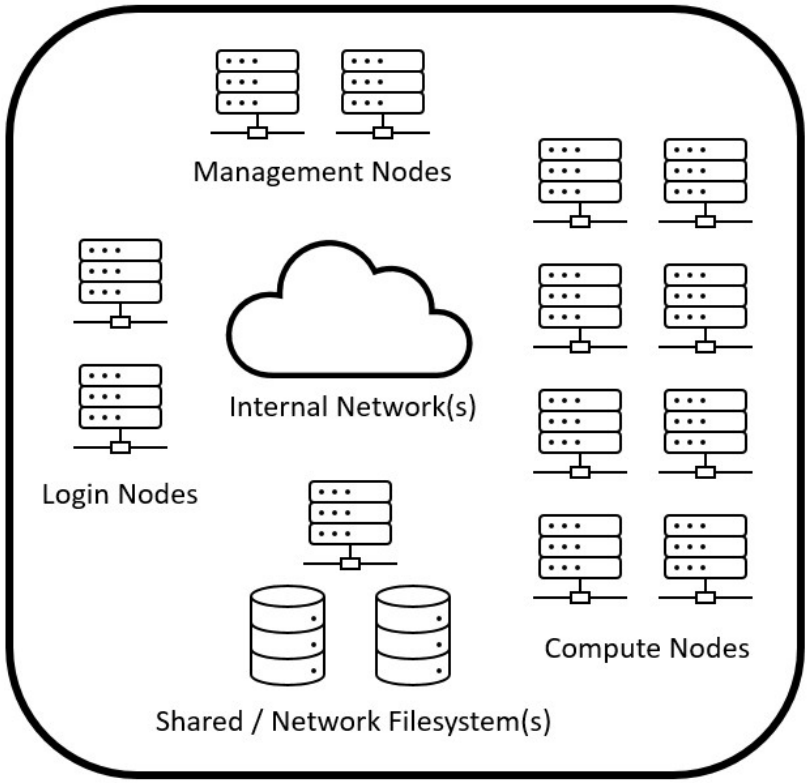
What is a batch job?

A batch job is a prescripted 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?

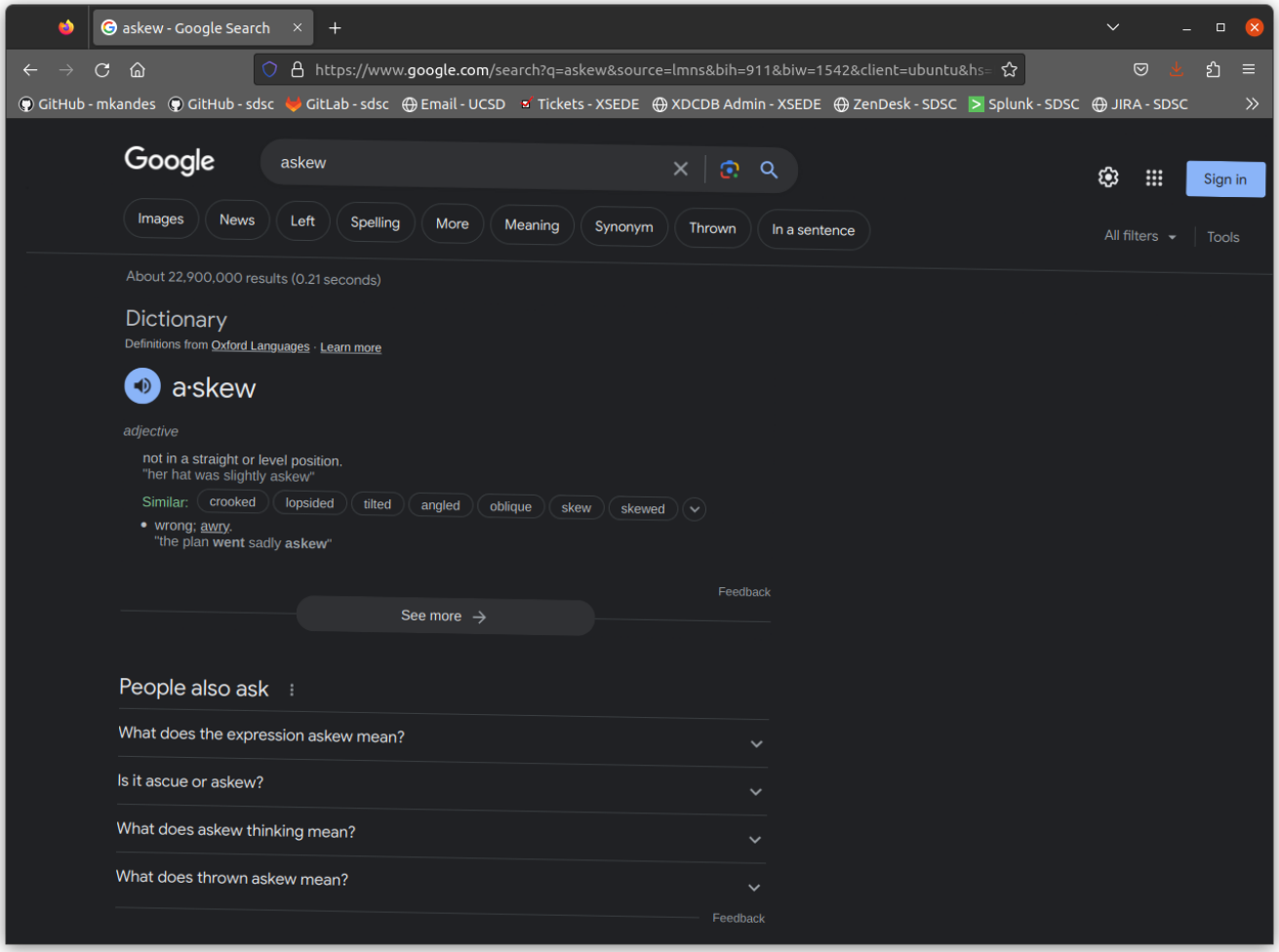
Slurm Workload Manager



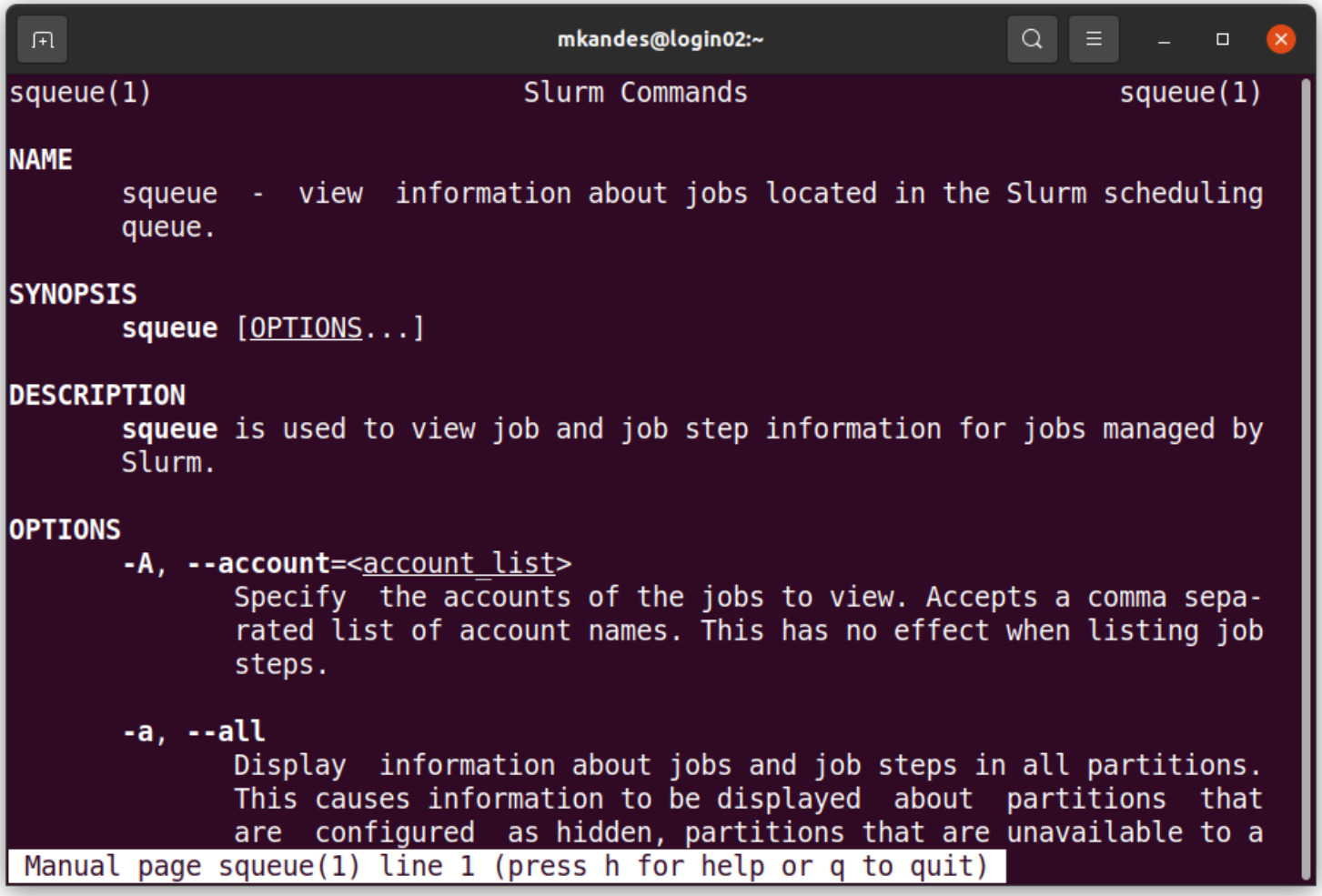


High-Performance Computing (HPC) System

Exercise 3: Looking squeue and beyond

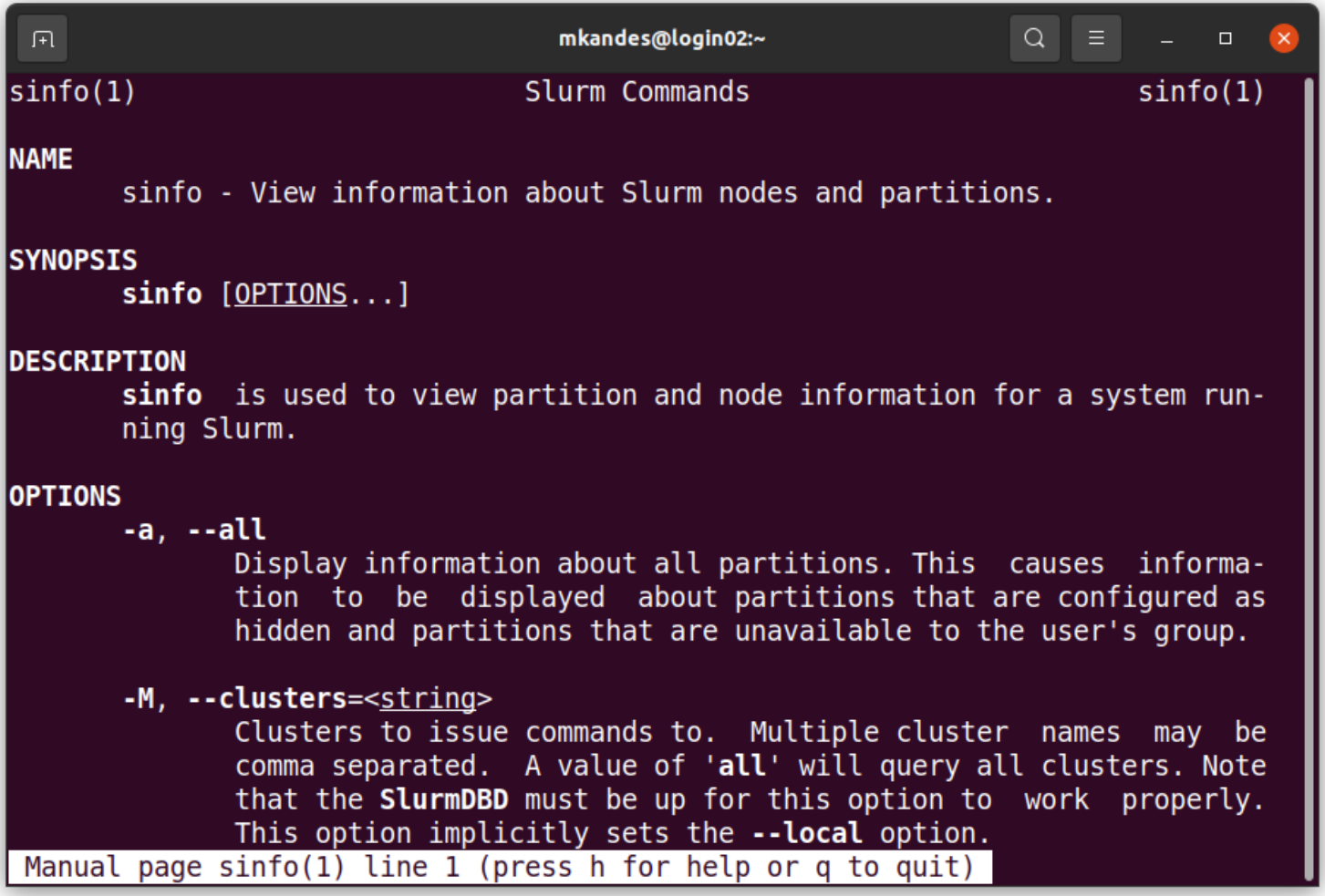


squeue




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

sinfo

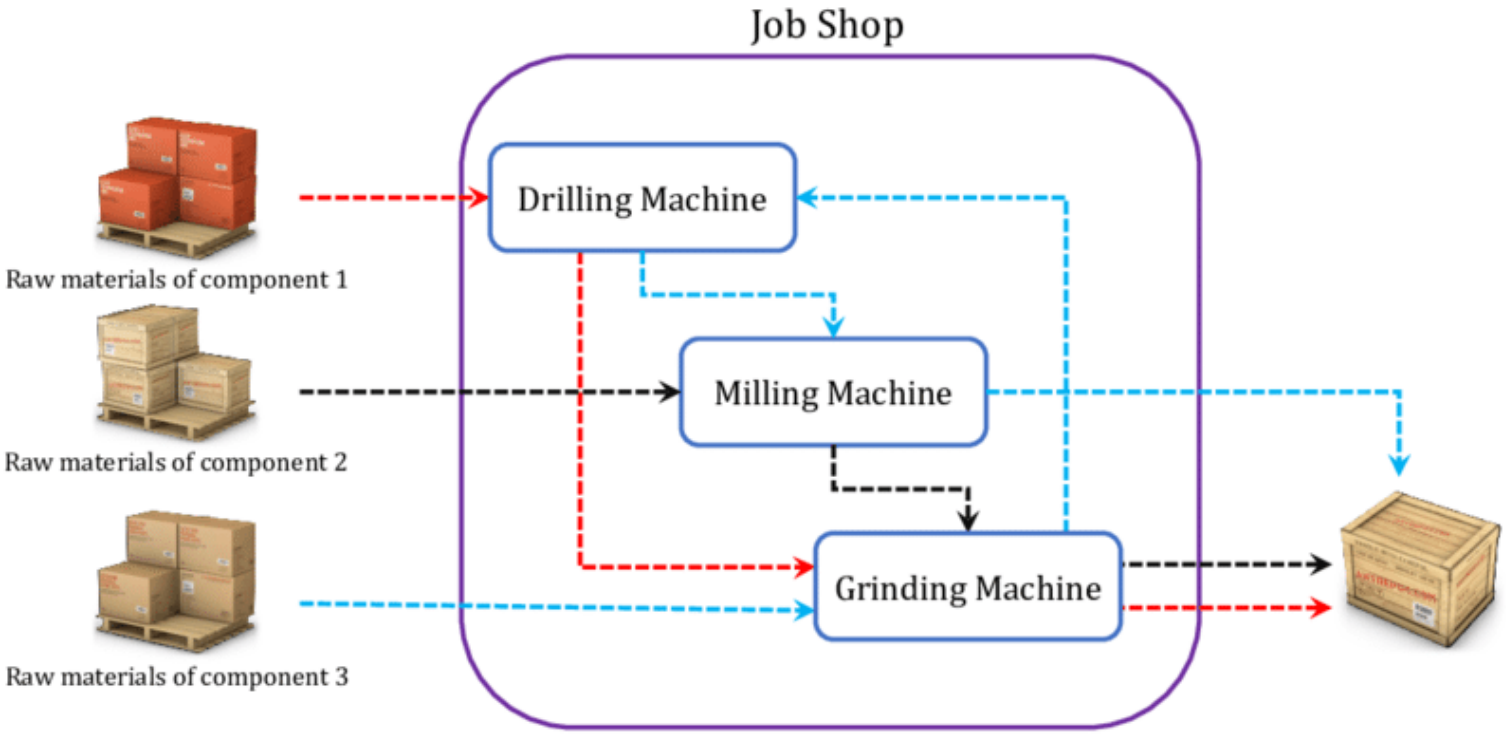

mkandes@login02:~
sinfo(1) Slurm Commands sinfo(1)
NAME
sinfo - View information about Slurm nodes and partitions.
SYNOPSIS
sinfo [OPTIONS...]
DESCRIPTION
sinfo is used to view partition and node information for a system running Slurm.
OPTIONS
-a, --all
Display information about all partitions. This 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]	llsubmit [script_file]
Job deletion	qdel [job_id]	scancel [job_id]	bkill [job_id]	qdel [job_id]	llcancel [job_id]
Job status (by job)	qstat [job_id]	squeue [job_id]	bjobs [job_id]	qstat -u * [-j job_id]	llq -u [username]
Job status (by user)	qstat -u [user_name]	squeue -u [user_name]	bjobs -u [user_name]	qstat [-u user_name]	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	qrls [job_id]	scontrol release [job_id]	bresume [job_id]	qrls [job_id]	llhold -r [job_id]
Queue list	qstat -Q	squeue	bqueues	qconf -sql	llclass
Node list	pbsnodes -l	sinfo -N OR scontrol show nodes	bhosts	qhost	llstatus -L machine
Cluster status	qstat -a	sinfo	bqueues	qhost -q	llstatus -L cluster
GUI	xpbsmon	sview	xlsf OR xlsbatch	qmon	xload
Environment	**PBS/Torque**	**Slurm**	**LSF**	**SGE**	**LoadLeveler**
Job ID	$PBS_JOBID	$SLURM_JOBID	$LSB_JOBID	$JOB_ID	$LOAD_STEP_ID
Submit Directory	$PBS_O_WORKDIR	$SLURM_SUBMIT_DIR	$LSB_SUBCWD	$SGE_O_WORKDIR	$LOADL_STEP_INITDIR
Submit Host	$PBS_O_HOST	$SLURM_SUBMIT_HOST	$LSB_SUB_HOST	$SGE_O_HOST	
Node List	$PBS_NODEFILE	$SLURM_JOB_NODELIST	$LSB_HOSTS/LSB_MCPU_HOST	$PE_HOSTFILE	$LOADL_PROCESSOR_LIST
Job Array Index	$PBS_ARRAYID	$SLURM_ARRAY_TASK_ID	$LSB_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/A	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]	-t [min] OR -t [days-hh:mm:ss]	-W [hh:mm:ss]	-l 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 stdout/err	-j oe (both to stdout) OR -j eo (both to stderr)	(use -o without -e)	(use -o without -e)	-j yes	
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]	--requeue OR --no-requeue (NOTE: configurable default)	-r	-r [yes|no]	restart=[yes|no]
Working Directory	N/A	--workdir=[dir_name]	(submission directory)	-wd [directory]	initialdir=[directory]
Resource Sharing	-l naccesspolicy=singlejob	--exclusive OR--shared	-x	-l exclusive	node_usage=not_shared
Memory Size	-l mem=[MB]	--mem=[mem][M|G|T] OR --mem-per-cpu=[mem][M|G|T]	-M [MB]	-l 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]		(Fixed allocation_rule in PE)	tasks_per_node=[count]
CPUs Per Task		--cpus-per-task=[count]			
Job Dependency	-d [job_id]	--depend=[state:job_id]	-w [done | exit | finish]	-hold_jid [job_id | job_name]	
Job Project		--wckey=[name]	-P [name]	-P [name]	
Job host preference		--nodelist=[nodes] AND/OR --exclude=[nodes]	-m [nodes]	-q [queue]@[node] OR -q [queue]@@[hostgroup]	
Quality Of Service	-l qos=[name]	--qos=[name]			
Job Arrays	-t [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		--licenses=[license_spec]	-R "rusage[license_spec]"	-l [license]=[count]	
Begin Time	-A "YYYY-MM-DD HH:MM:SS"	--begin=YYYY-MM-DD[THH:MM[:SS]]	-b[[year:][month:]daty:]hour:minute	-a [YYMMDDhhmm]	

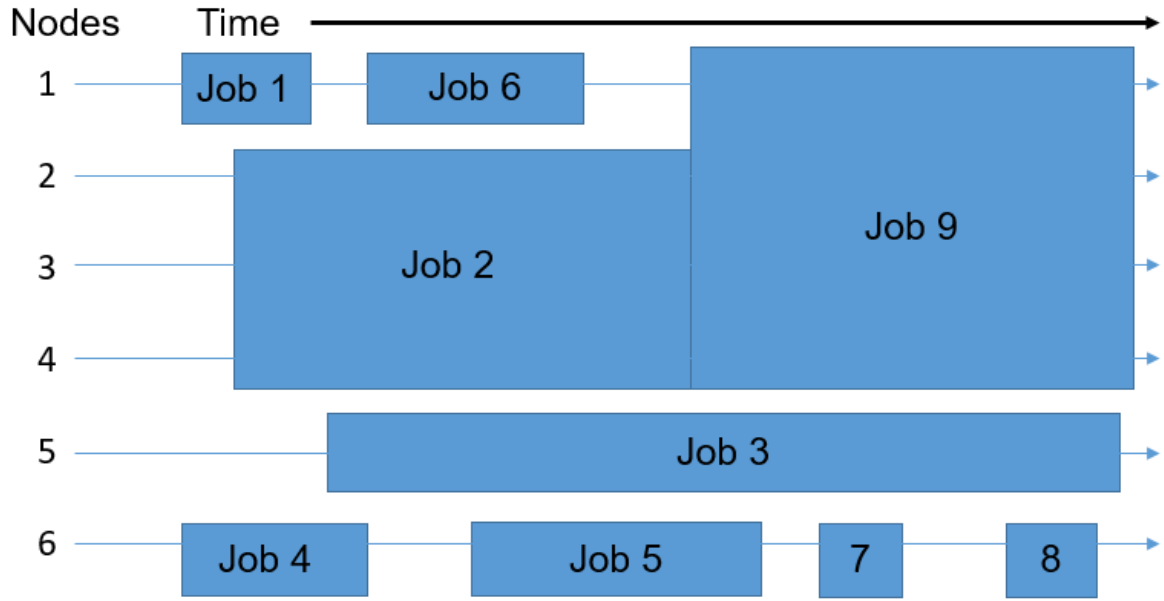
https://slurm.schedmd.com/rosetta.html

How does a scheduler work?



See: Job-Shop Scheduling Problem

How does a scheduler work?

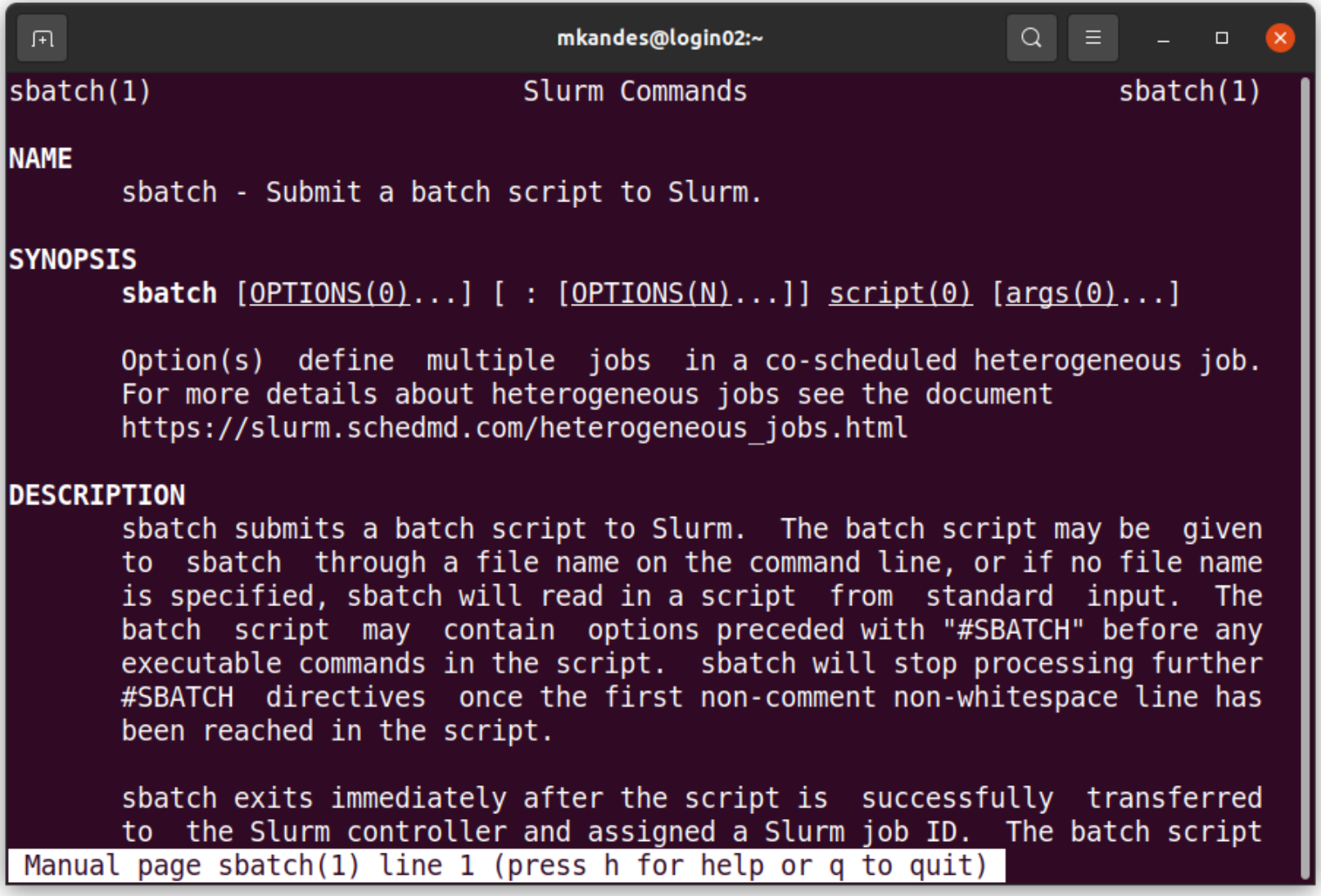


In general, every scheduler has three main goals:

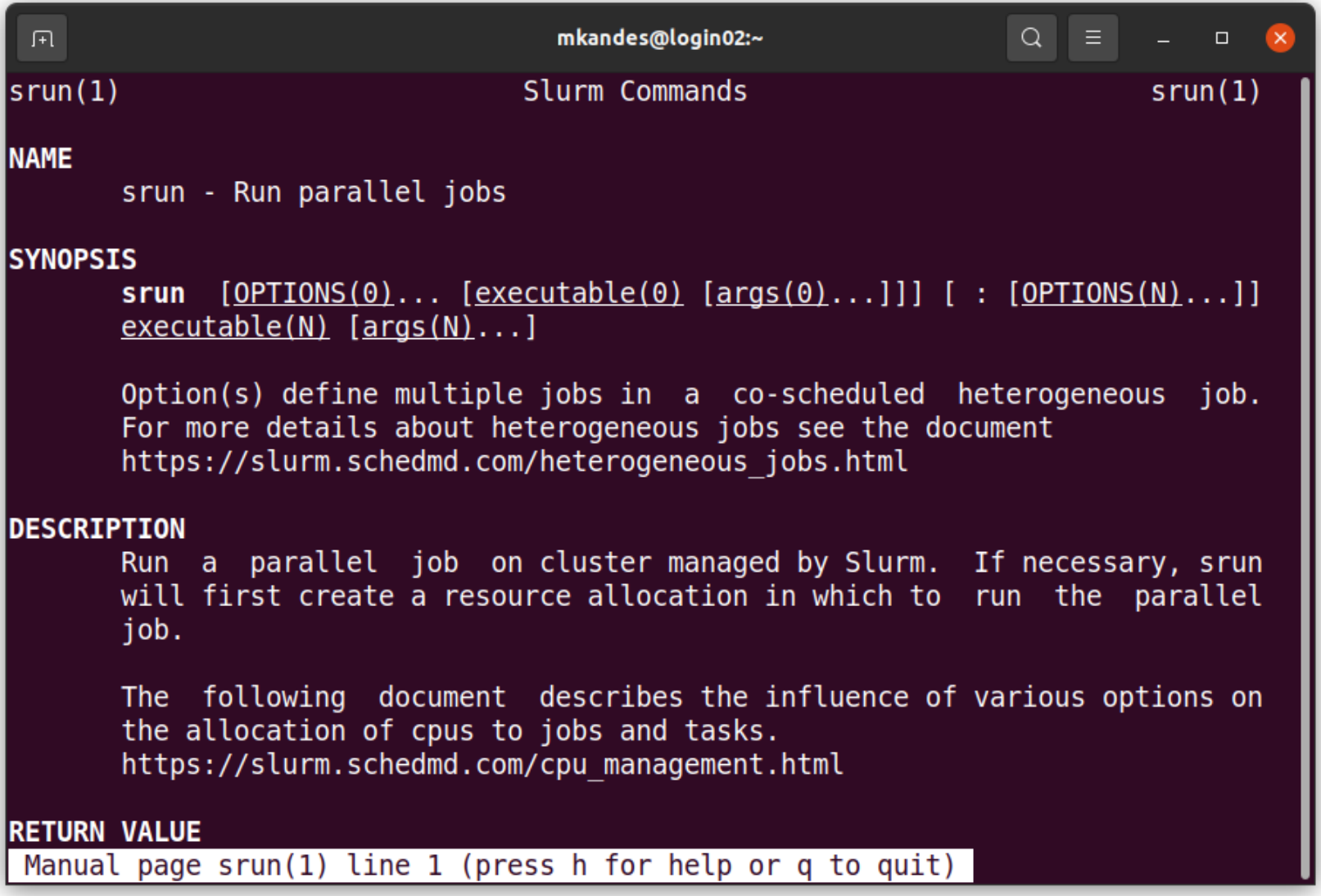
- Minimize the time between the job submission and finishing the job: no job should stay in the queue for extensive periods of time
- Optimize CPU utilization: the CPUs of the supercomputer are one of the core resources for a big application; therefore, there should only be few time slots where a CPU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

Exercise 4: Hello, My Hostname Is

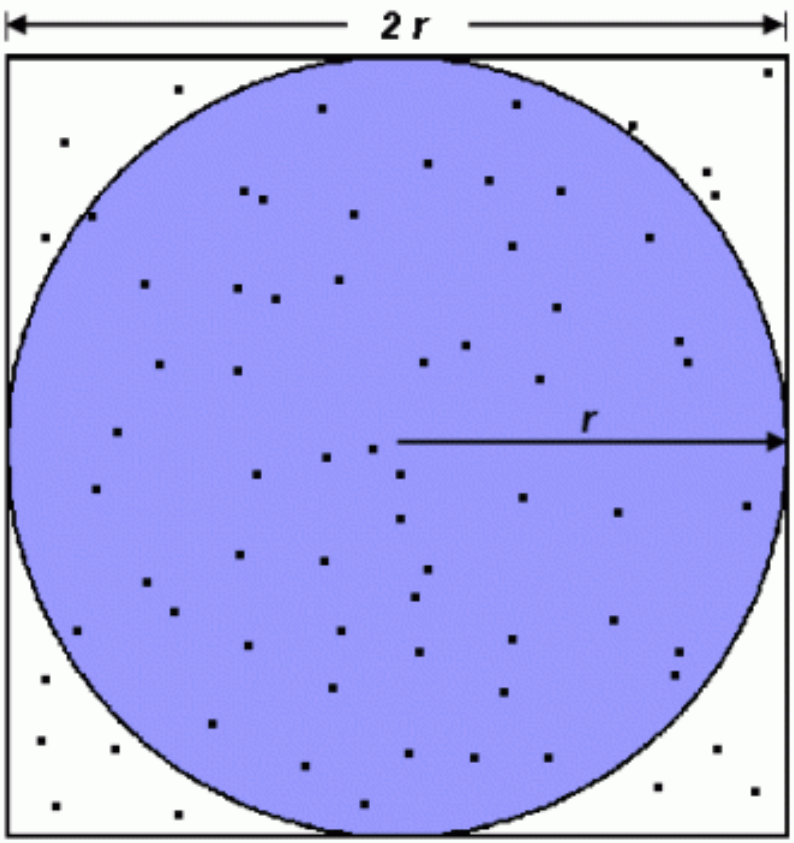
sbatch


mkandes@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

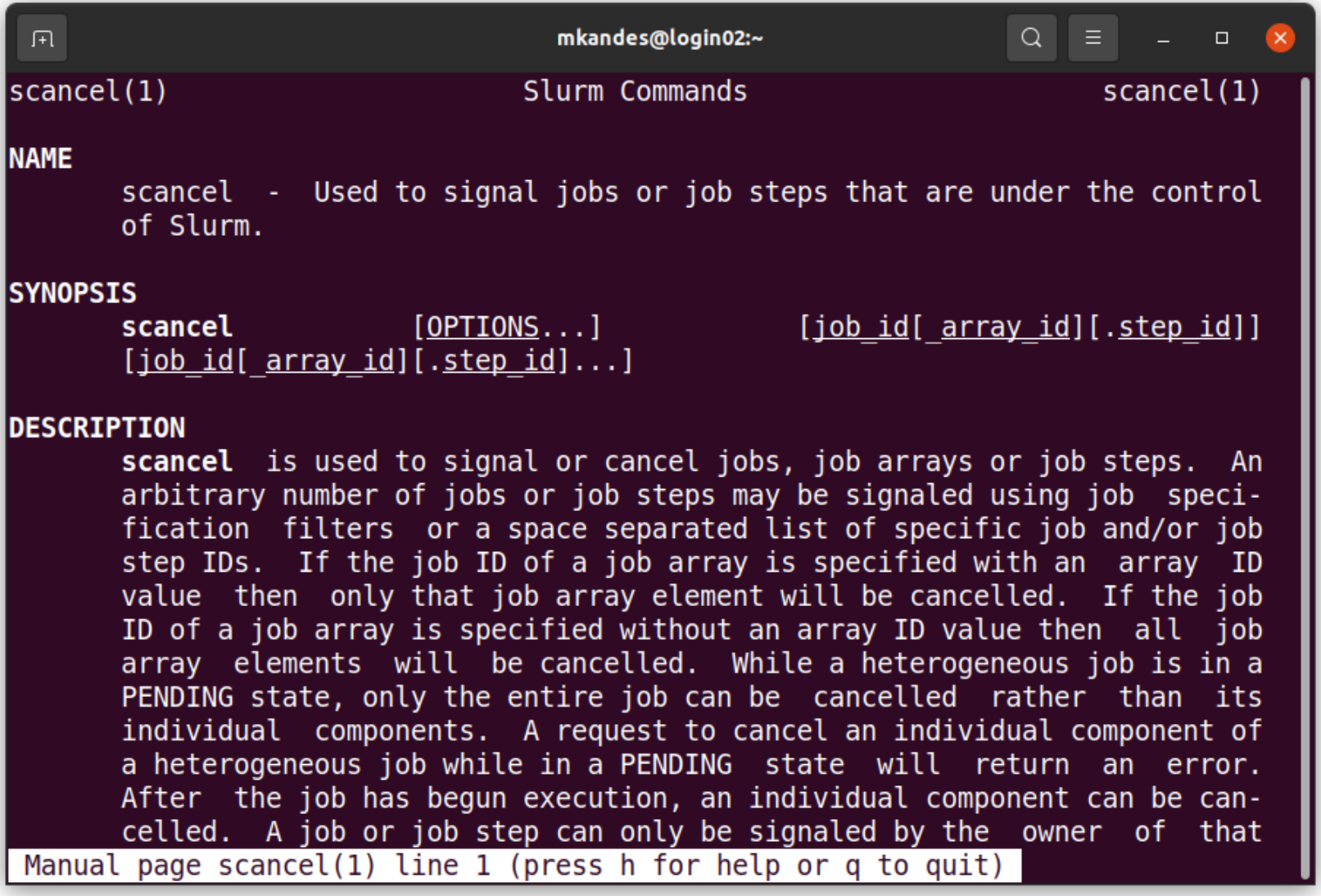


$$A_S = (2r)^2 = 4r^2$$

$$A_C = \pi r^2$$

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

scancel


mkandes@login02:~
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 specification 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 cancelled. 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?

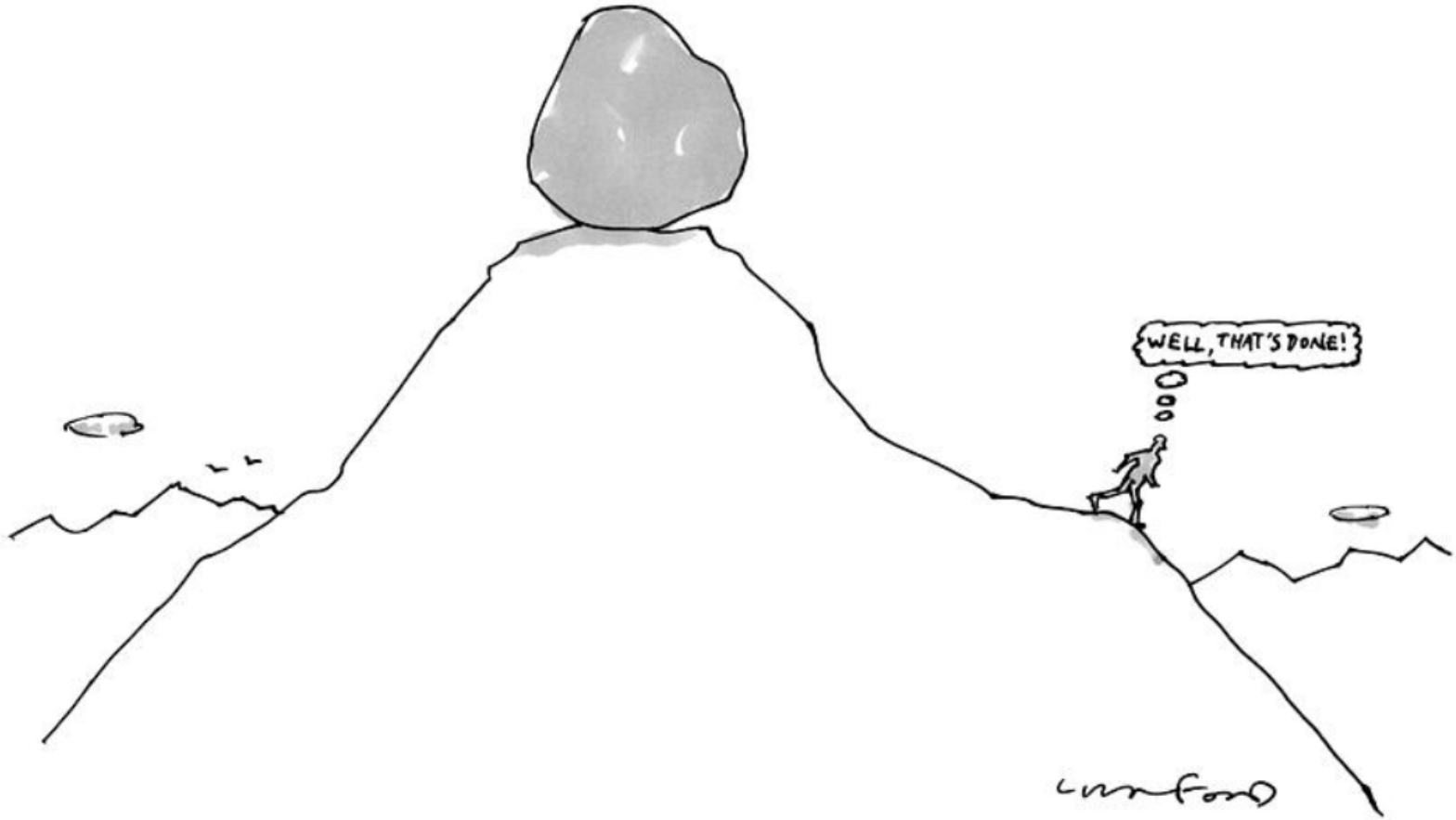


Image Credit: New Yorker - M. Crawford