SDSC Summer Institute
Day 1: Batch Jobs

July 27, 2022

Mary Thomas

San Diego Supercomputer Center

SDSC SAN DIEGO SUPERCOMPUTER CENTER
UC San Diego

Outline

- Introduction to Batch Jobs
- What is a Batch Scheduler
- Anatomy of a Batch Script
- Using the SLURM Environment
- Using Batch Scripts on Expanse

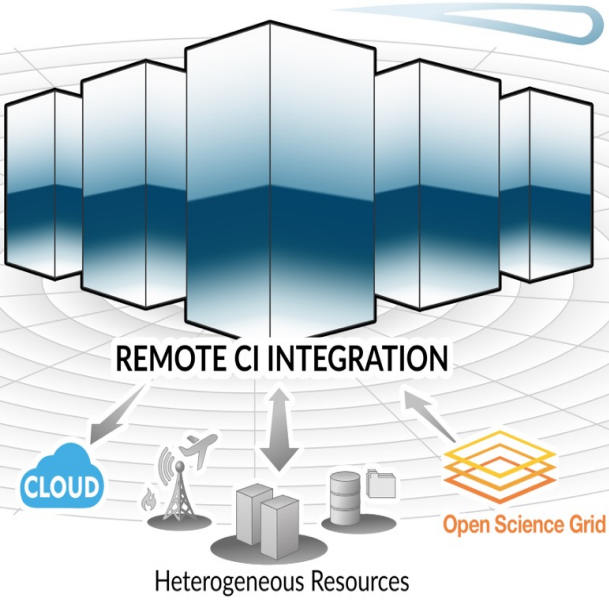
Expanse

EXPANSE

COMPUTING WITHOUT BOUNDARIES
5 PETAFLOP/S HPC and DATA RESOURCE

HPC RESOURCE
13 Scalable Compute Units
728 Standard Compute Nodes
52 GPU Nodes: 208 GPUs
4 Large Memory Nodes

DATA CENTRIC ARCHITECTURE
12PB Perf. Storage: 140GB/s, 200k IOPS
Fast I/O Node-Local NVMe Storage
7PB Ceph Object Storage
High-Performance R&E Networking



LONG-TAIL SCIENCE
Multi-Messenger Astronomy
Genomics
Earth Science
Social Science

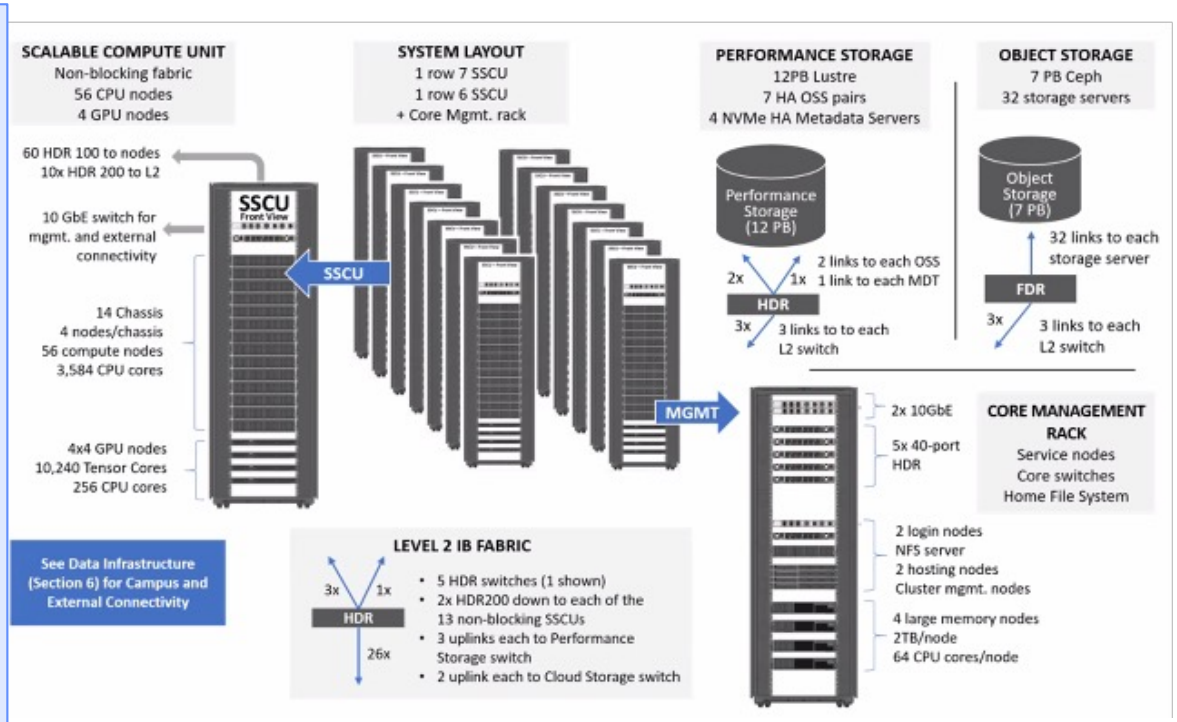
INNOVATIVE OPERATIONS
Composable Systems
High-Throughput Computing
Science Gateways
Interactive Computing
Containerized Computing
Cloud Bursting

For more details see the Expanse user guide @ https://www.sdsc.edu/support/user_guides/expanse.html
and the "Introduction to Expanse" webinar @ https://www.sdsc.edu/event_items/202006_Introduction_to_Expanse.html

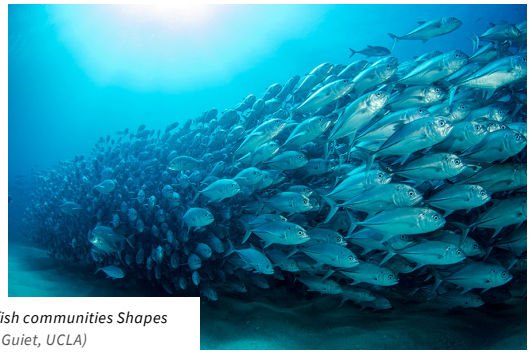
Expanse Heterogeneous Architecture

System Summary

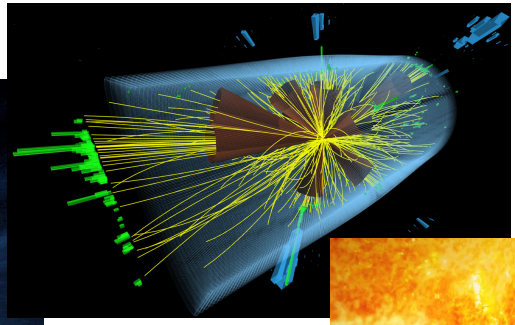
- *13 SDSC Scalable Compute Units (SSCU)*
- *728 x 2s Standard Compute Nodes*
- *93,184 Compute Cores*
- *200 TB DDR4 Memory*
- *52x 4-way GPU Nodes w/NVLINK*
- *208 V100s*
- *4x 2TB Large Memory Nodes*
- *HDR 100 non-blocking Fabric*
- *12 PB Lustre High Performance*
- *Storage*
- *7 PB Ceph Object Storage*
- *1.2 PB on-node NVMe*
- *Dell EMC PowerEdge*
- *Direct Liquid Cooled*



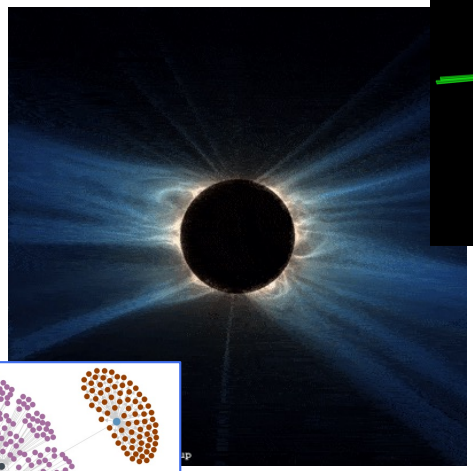
You can run Amazing Jobs on Supercomputers!



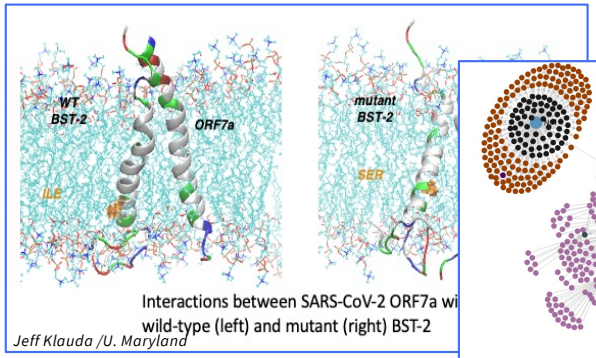
Pelagic fish communities Shapes (Jerome Guiet, UCLA)



Compact Muon Solenoid (**CMS**)experiment at the LHC, CERN. *Image courtesy of CMS Collaboration; Mc Cauley, Thomas*

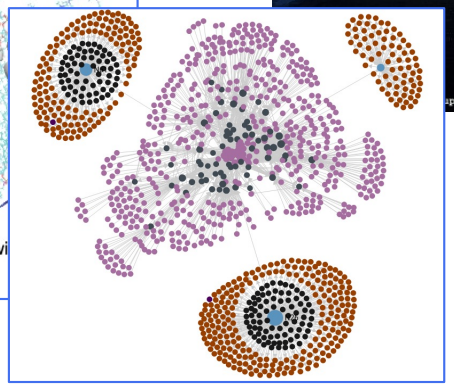


Model of Both Inner and Outer Solar System M. S.Clement (Carnegie Institution for Science)



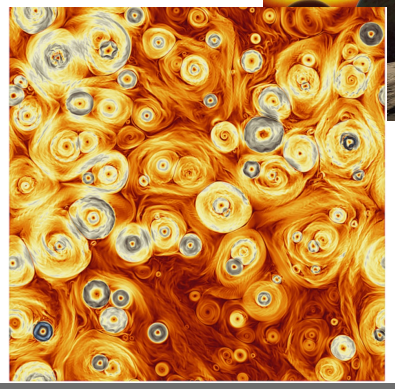
Interactions between SARS-CoV-2 ORF7a wi
wild-type (left) and mutant (right) BST-2

Jeff Klauda /U. Maryland



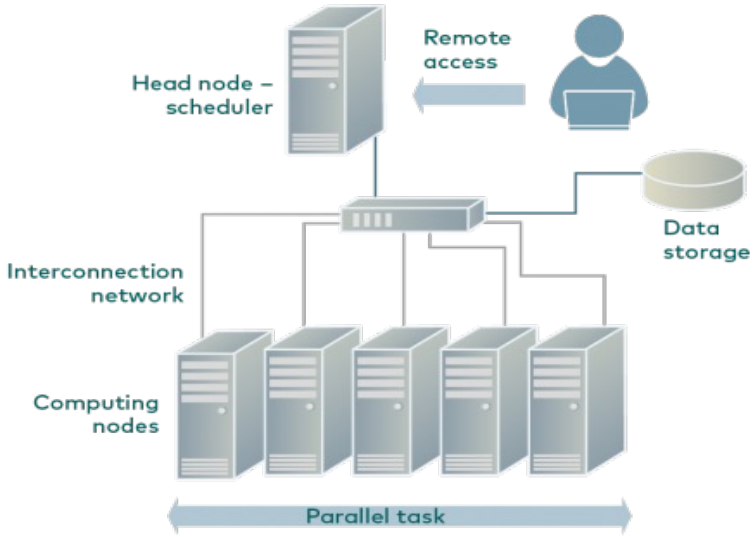
Cooper Downs, Predictive Science Inc

Sample of Internet structure from CAIDA data (Mark Burgess, 12/16/21)



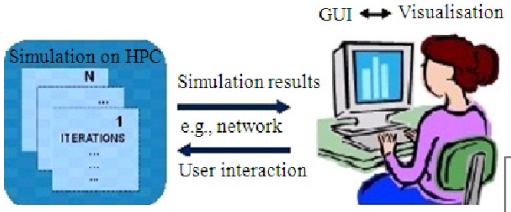
Modeling the sun's corona, Alfred Mallet (UC Berkeley)

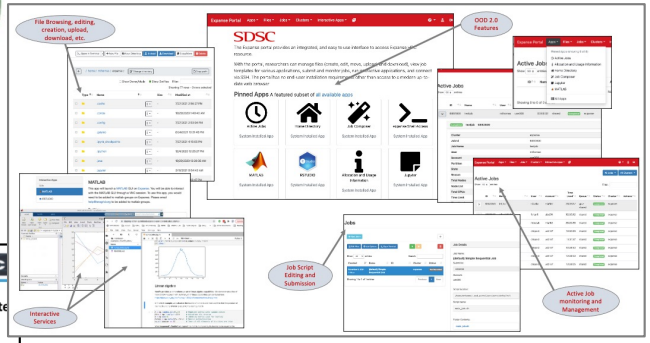
But You Have to Run Them on HPC Clusters

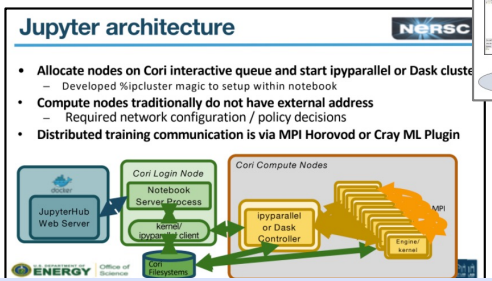


- Jobs can be run from:
  - the command line
  - Application clients
  - The batch queue system
- Jobs can have different sizes:
  - From 1 core → 1000s of nodes
- For large jobs, schedulers are needed to coordinate jobs
- This can be non-trivial

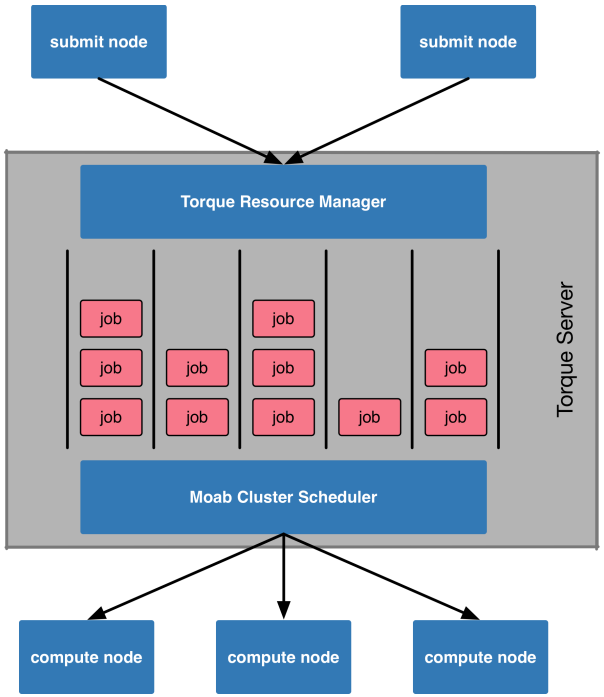
Jobs Can be Run Interactively or as Batch Jobs (Background)







Interactive Distributed Computing with Jupyter (NERSC)



Src: https://hpc.dccn.nl/_images/torque_moab_arch.png

Interactive HPC Computing

See SDSC DI22: Session 3.2 Interactive Computing

- In **computer** science, **interactive computing** refers to software which accepts input from the user as it runs.
  - **Interactive** software includes commonly used programs, such as word processors or spreadsheet applications.
- **Interactive HPC computing** involves *real-time* user inputs to perform tasks on a set of compute node(s) including:
  - Code development, real-time data exploration, and visualizations.
  - Used when applications have large data sets or are too large to download to local device, software is difficult install, etc.
  - User inputs come via command line interface or application GUI (Jupyter Notebooks, Matlab, R-studio).
  - Actions performed on remote compute nodes as a result of user input or program out.

Interactive CPU node: Command Line

```
[username@login01 openmp]$ module purge
[username@login01 openmp]$ module load slurm
[username@login01 openmp]$ module load cpu
[username@login01 openmp]$ module load gcc/10.2.0
[username@login01 openmp]$ module load openmpi/4.0.4
[username@login01 openmp]$ srun --partition=debug  --pty --account=use300--nodes=1 --ntasks-per-node=64  --
mem=128G -t 00:30:00 --wait=0 --export=ALL /bin/bash
```

Request an interactive node for 30 minutes

```
[mthomas@exp-9-55 openmp]$ export OMP_NUM_THREADS=16
[mthomas@exp-9-55 openmp]$ ./hello_openmp
 HELLO FROM THREAD NUMBER =       0
 HELLO FROM THREAD NUMBER =       8
 HELLO FROM THREAD NUMBER =       9
 HELLO FROM THREAD NUMBER =       10
 HELLO FROM THREAD NUMBER =       11
 HELLO FROM THREAD NUMBER =       12
 HELLO FROM THREAD NUMBER =       13
 HELLO FROM THREAD NUMBER =       14
 HELLO FROM THREAD NUMBER =       15
 HELLO FROM THREAD NUMBER =       4
 HELLO FROM THREAD NUMBER =       7
 HELLO FROM THREAD NUMBER =       6
 HELLO FROM THREAD NUMBER =       5
 HELLO FROM THREAD NUMBER =       3
 HELLO FROM THREAD NUMBER =       2
 HELLO FROM THREAD NUMBER =       1
```

Computation is done on the node and output displayed in real time

- Exit interactive session when your work is done or you will be charged CPU time.
- Beware of oversubscribing your job: don't ask for more cores than you have requested.
- Intel compiler allows this, but your performance will be degraded.

Outline

- Introduction to Batch Jobs
- What is a Batch Scheduler
- Anatomy of a Batch Script
- Using the SLURM Environment
- Using Batch Scripts on Expanse

What is a Scheduler & How are They Used?

- Any HPC (or HTC) system — usually a cluster of machines — needs a means of sharing computational resources fairly between users; without, there would be anarchy.
- Batch-queueing systems — usually abbreviated to simply *batch systems* — are intended to do this.
- All batch systems have at least these features:
  - *a scheduler* for allocating resources (CPUs!) to jobs and for prioritising jobs;
  - *one or more queues* to which jobs are submitted
  - note: each queue might be configured for a particular type of job, for example, serial or parallel jobs, long or short jobs, or those requiring particularly high memory. These are called partitions (or job queues)

Batch Scheduler Main Goals

- Minimize time between job submission and completion:
  - No job should stay in queue for extensive periods of time.
- Optimize CPU utilization:
  - Algorithms focus on minimizing CPU idle times.
- Maximize job throughput:
  - Manage as many jobs per time unit as possible.
- Support running jobs automatically in the background

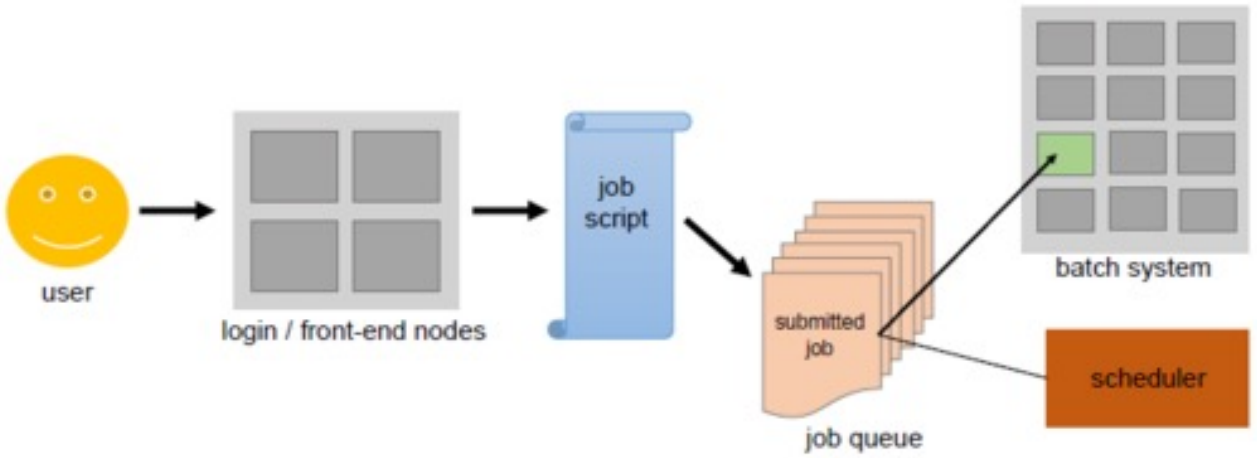
Common Batch Scheduler Terms

- batch processing comes from the idea of batch production (production of a "batch" of multiple items at once, one stage at a time)
- Batch job scheduler: computer application for controlling unattended background execution of jobs.
- batch scheduling: execution of non-interactive jobs
- batch processing: method of running software programs called jobs in batches automatically.

- Typically, users required to submit the jobs, no other interaction by the user is required to process the batch.
- Batches may automatically be run at scheduled times as well as being run contingent on the availability of computer resources
- The data structure of jobs to run is known as the job queue.
- Synonyms include: batch system, distributed resource management system (DRMS), distributed resource manager (DRM), workload automation (WLA).

https://en.wikipedia.org/wiki/Job_scheduler

Simple “Batch Scheduler” Architecture



- Batch scheduler: software that implements a *batch system* on a cluster.
- Users do not run calculations interactively -- instead they submit *non-interactive batch jobs* to the *scheduler*.
- All work about the same: some are open source; some cost money; some are very expensive.

Common Cluster Software

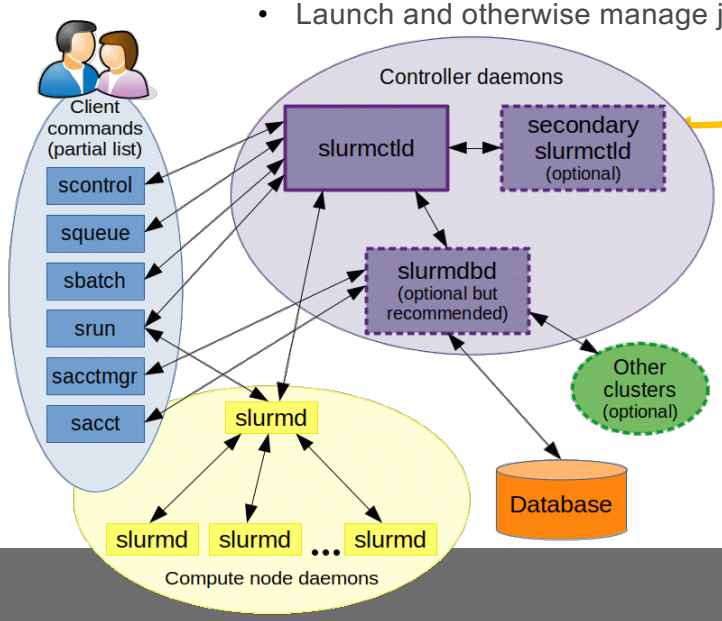
Software	Maintainer	ArchitectureOCS	High-Performance/ High-Throughput Computing	License	Platforms supported
Accelerator	Altair	Master/worker distributed	HPC/HTC	Proprietary	Linux, Windows
Grid MP	Univa (formerly United Devices)	Distributed master/worker	HTC/HPC	Proprietary	Windows, Linux, Mac OS X, Solaris
Moab Cluster Suite	Adaptive Computing		HPC	Proprietary	Linux, Mac OS X, Windows, AIX, OSF/Tru-64, Solaris, HP-UX, IRIX, FreeBSD & other UNIX platforms
OpenLava	Teraproc	Master/Worker, multiple admin/submit nodes	HTC/HPC	GPL	Linux
PBS Pro	Altair	Master/worker distributed with fail-over	HPC/HTC	AGPL or Proprietary	Linux, Windows
SLURM	SchedMD		HPC/HTC	GPL	Linux/*nix
Spectrum LSF	IBM	Master node with failover/exec clients, multiple admin/submit nodes, Suite addOns	HPC/HTC	Proprietary	Unix, Linux, Windows

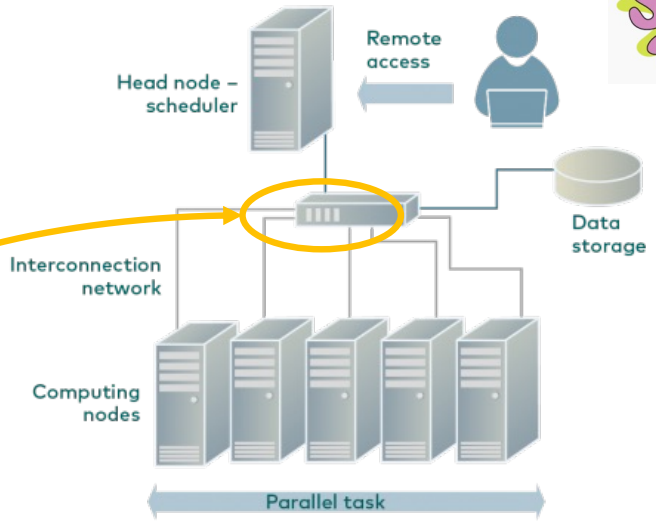
Source: https://en.wikipedia.org/wiki/Comparison_of_cluster_software

Batch Jobs: Slurm Resource Manager

Simple Linux Utility for Resource Management

- Open Source, runs on many systems
- “Glue” for parallel computer to schedule and execute jobs
- Role: Allocate resources within a cluster
  - Nodes (unique IP address)
  - Interconnect/switches
  - Generic resources (e.g. GPUs)
  - Launch and otherwise manage jobs



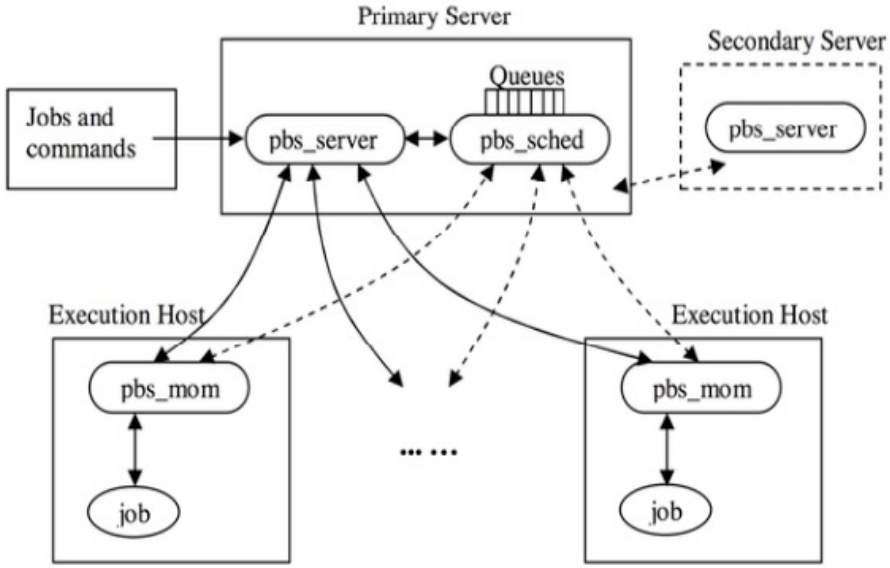


- Functionality:
  - Prioritize queue(s) of jobs;
  - decide when and where to start jobs;
  - terminate job when done;
  - Appropriate resources;
  - manage accounts for jobs

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

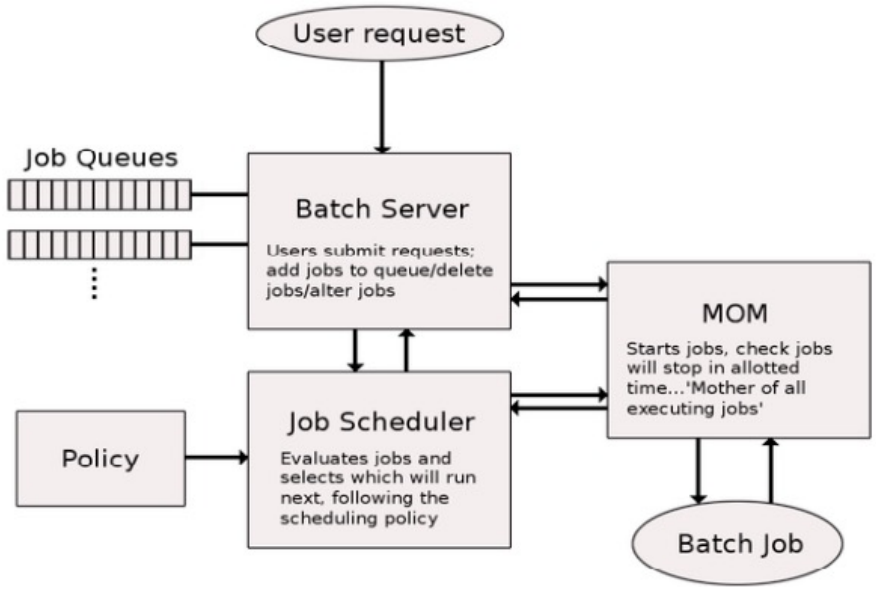
Portable Batch Scheduler (PBS/TORQUE)

- Distributed resource manager providing control over batch jobs and distributed compute nodes.
- PBS/TORQUE can integrate with the non-commercial Maui Cluster Scheduler or the commercial Moab Workload Manager to improve overall utilization, scheduling and administration on a cluster.



PBS Job Server & Components

- PBS Job Server: commands/daemons communicate with Server
  batch job services: receiving/creating, running, modifying, protecting against system crashes
- PBS Job Scheduler: controls when/where jobs run communicate with machine oriented mini-server (MOM)
- PBS MOM (Machine Oriented Miniserver): Starts job, makes sure it completes in specified time user login session
- PBS Client Programs : command line or GUI, user, operator, manager submit, monitor, modify, delete



PBS/TORQUE Job life cycle

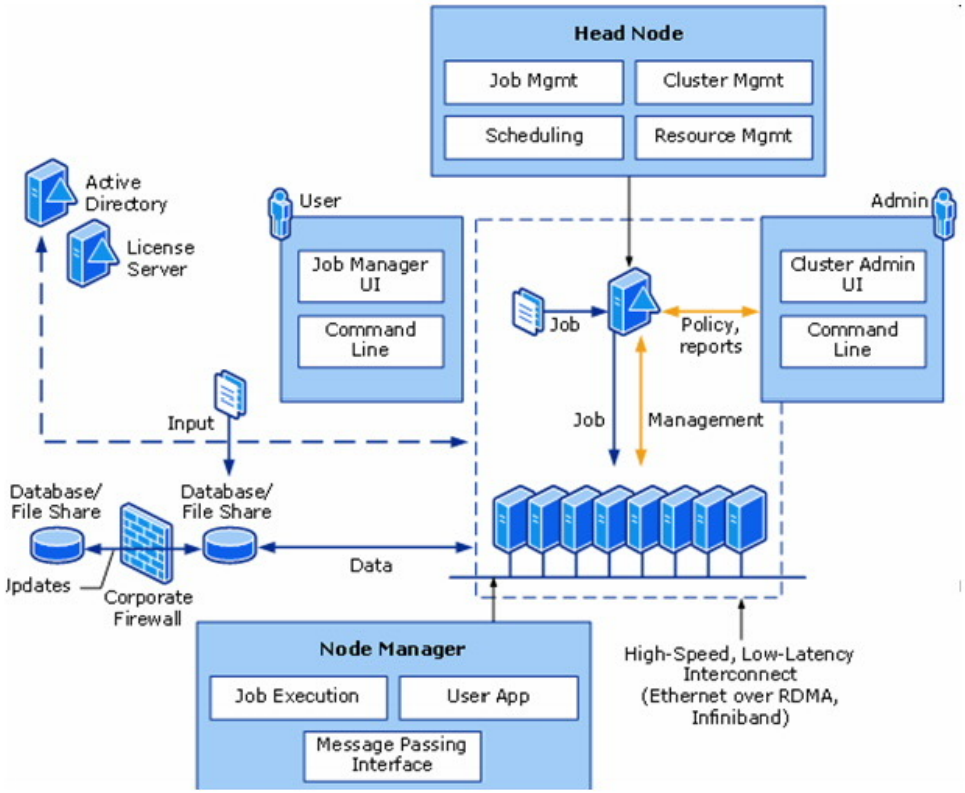
Stage	Description

Creation	Typically, a submit script is written to hold all of the parameters of a job. These parameters could include how long a job should run (**walltime**), what resources are necessary to run, and what to execute. The following is an example submit file:
	`#PBS -N localBlast`
	`#PBS -S /bin/sh`
	`#PBS -l nodes=1:ppn=2,walltime=240:00:00`
	`#PBS -M user@my.organization.com`
	`#PBS -m ea`
	`source ~/.bashrc`
	`cd $HOME/work/dir`
	`sh myBlast.sh -i -v`
	This submit script specifies the name of the job (localBlast), what environment to use (`/bin/sh`), that it needs both processors on a single node (**nodes=1:ppn=2**), that it will run for at most 10 days, and that TORQUE should email "user@my.organization.com" when the job exits or aborts. Additionally, the user specifies where and what to execute.

Submission	A job is submitted with the qsub command. Once submitted, the policies set by the administration and technical staff of the site dictate the priority of the job and therefore, when it will start executing.
Execution	Jobs often spend most of their lifecycle executing. While a job is running, its status can be queried with qstat.
Finalilzation	When a job completes, by default, the `stdout` and `stderr` files are copied to the directory where the job was submitted.

Spectrum Load Shaing Facility (LSF)

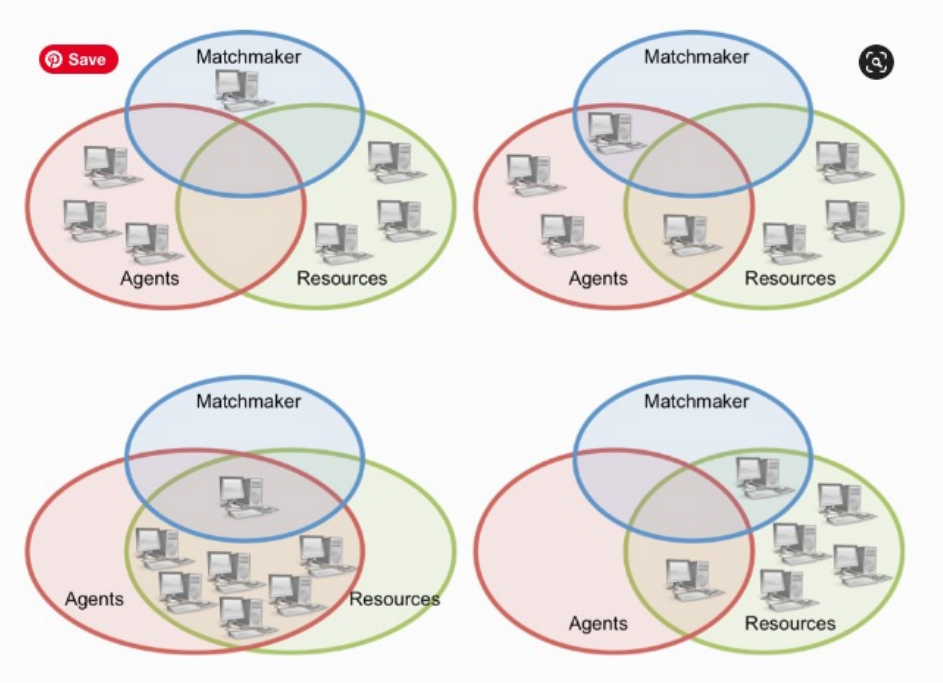
- Spectrum LSF designed to support diverse distributed workloads.
- Shares resources based on flexible policies.
- Supports serial and parallel batch jobs along with a variety of other application models.
  - These include interactive workloads
  - parametric/array jobs
  - multi-step workflows, virtualized and containerized workloads
  - “long-running” distributed services such as TensorFlow, Spark, or Jupyter notebooks.
- Runs on IBM & Linux; Proprietary



https://en.wikipedia.org/wiki/IBM_Spectrum_LSF

HTCondor

- HTCondor is a job scheduling and resource management software.
- HTCondor creates an HTC system by grouping, or “pooling”, network-connected computing resources.
- It can be used to manage workloads on a dedicated cluster of computers, or to farm out work to idle desktop computers – so-called cycle scavenging.

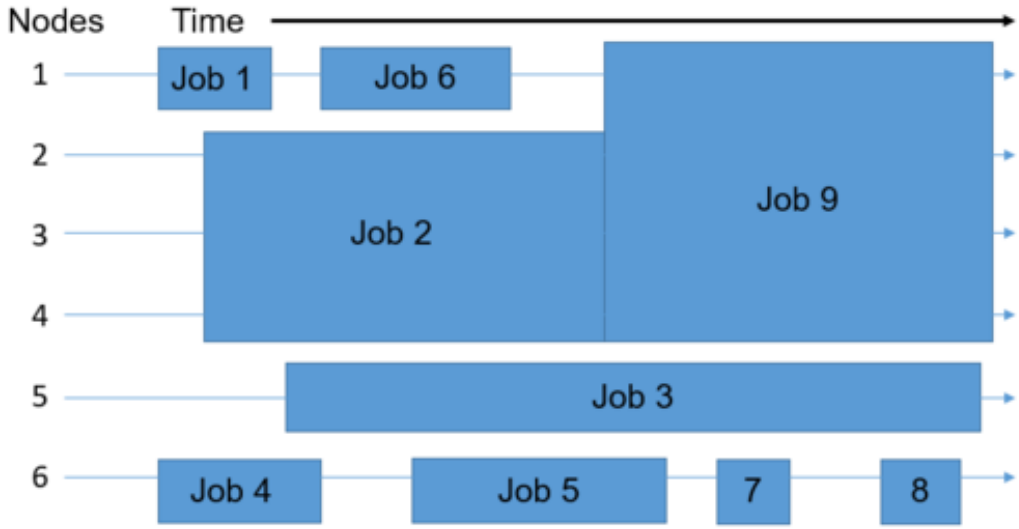


Possible HTCondor pool configurations.

How a Scheduler Schedules Jobs

- Simple example:
  - a 6 node system
  - yser wants to run 9 jobs.
- Scheduler places the jobs in the queue and then onto the available nodes as they open up.

Many parameters affect scheduling: number of jobs submitted, required runtime, required number of cores, required main memory, accelerators, libraries, etc.



Scheduler needs to play kind of "multidimensional tetris" to fill the cluster's nodes evenly and efficiently.

https://hpc-wiki.info/hpc/Scheduling_Basics

Common Scheduling Algorithms (simple)

First Come, First Serve

- Jobs are run in exact same order in which they first enter the queue.
- Advantage is that every job will definitely be run.
- However, very tiny jobs might wait for an inadequately long time compared to their actual execution time.

Shortest Job First

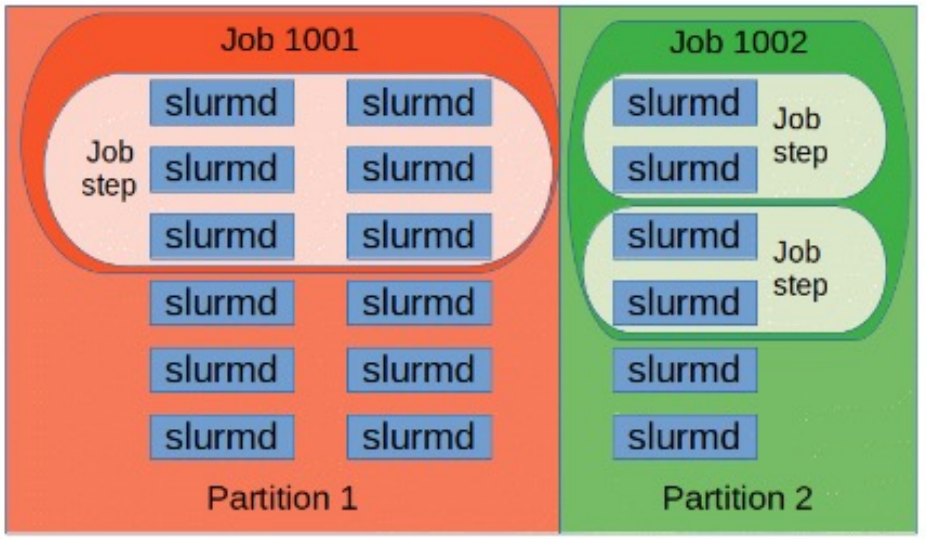
- Based on the execution time declared in the jobscript, the scheduler can estimate how long it will take to execute the job.
- Then, the jobs are ranked by that time from shortest to longest.
- While short jobs will start after a short waiting time, long running jobs (or at least jobs declared as such) might never actually start.

Backfilling

- Scheduler maintains concept of "First Come, First Serve" without preventing long running jobs to execute.
- Scheduler checks whether first job in queue can be executed:
  - If true, job is executed without further delay.
  - If not, scheduler looks for next job that can be executed without extending the waiting time of the first job in queue and runs it.
- Jobs that only need a few compute resources, are easily "backfillable,"
  - small jobs will usually encounter shorter queue times

Partitions (SLURM example)

- Group nodes into logical (possibly overlapping) sets
- Can be considered job queues, each of which has an assortment of constraints such as job size limit, job time limit, users permitted to use it, etc.
- ***sinfo*** reports the state of partitions and nodes managed by Slurm.
  - It has a wide variety of filtering, sorting, and formatting options.



Outline

- Introduction to Batch Jobs
- What is a Batch Scheduler
- Anatomy of a Batch Script
- Using the SLURM Environment
- Using Batch Scripts on Expanse

Batch Scripts -- Used to Launch Jobs

- Batch Jobs: Submit batch scripts from the login nodes to a batch service:
  - Expanse uses the Simple Linux Utility for Resource Management (SLURM) batch environment.
- Can be used to run serial jobs (1 core), multi-threaded (OpenMP), multi-core jobs, and parallel (multi-node, MPI) jobs
- Parameters you can set:
  - Partition (queue)
  - Time limit for the run (maximum of 48 hours)
  - Number of nodes, tasks per node; Memory requirements (if a
  - You can define the job name, output file location; email info, c n

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

Batch Scripts Contents

```
[uswername@login02 calc-prime]$ cat mpi-prime-slurm.sb
#!/bin/bash
#SBATCH --job-name="mpi_prime"
#SBATCH --output="mpi_prime.%j.%N.out"
####SBATCH --partition=compute
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --export=ALL
#SBATCH -t 00:10:00
#SBATCH -A use300

## Environment
module purge
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4

## echo job name and id:
echo "SLURM_JOB_NAME:  $SLURM_JOB_NAME"
echo "SLURM_JOB_ID:    $SLURM_JOB_ID"
d=`date`
echo "DATE:  $d"
```

Batch Script 4 Main Parts:

- “Shebang” !
  - #!/bin/bash command tells the shell to run the script using the bash
- Resource Request:
  - options preceded with "#SBATCH" before any executable commands in the script
- Dependencies
  - Loads software that the project depends on to execute
- Job Steps:
  - Specify the list of tasks to be carried out.

https://hpc-wiki.info/mediawiki/hpc_images/b/bb/Scheduler.PNG

Batch Script Output: ENV_Info

```
[uswername@login02 env_info ]$ cat env-slurm.sb
#!/bin/bash
#SBATCH --job-name=" env_info "
#SBATCH --output="mpi_prime.%j.%N.out"
####SBATCH --partition=compute
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --export=ALL
#SBATCH -t 00:10:00
#SBATCH -A use300

## Environment
module purge
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4

## echo job name and id:
echo "SLURM_JOB_NAME:  $SLURM_JOB_NAME"
echo "SLURM_JOB_ID:    $SLURM_JOB_ID"
d=`date`
echo "DATE:  $d"
e=`env`
echo "env= $e"
```

```
[mthomas@login02 env_info ]$ sbatch env-slurm.sb
Submitted batch job 14126259
[mthomas@login02 env_info]$ !sq
squeue -u mthomas
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
14126259     debug env_info   mthomas  R       0:04      1 exp-9-55
[mthomas@login01 env_info ]$ cat env_info.14126259.exp-4-35.out
SLURM_JOB_NAME:  env_info
SLURM_JOB_ID:    14126259
hostname=  exp-4-35
date=  Sun Jun 26 22:05:15 PDT 2022
whoami=  mthomas
pwd=  /home/mthomas/hpctr-examples/env_info
Currently Loaded Modules:       1) slurm/expanse/21.08.8   2) cpu/0.15.4
---------------------------------
env= LD_LIBRARY_PATH=/cm/shared/apps/slurm/current/lib64/slurm:
[SNIP]
/cm/shared/apps/slurm/current/lib64
SLURM_SUBMIT_DIR=/home/mthomas/hpctr-examples/env_info
HISTCONTROL=ignoredups
DISPLAY=localhost:16.0
HOSTNAME=exp-4-35
[SNIP]
```

Outline

- Introduction to Batch Jobs
- What is a Batch Scheduler
- Anatomy of a Batch Script
- Using the SLURM Environment
- Using Batch Scripts on Expanse

Basic Job management

- **squeue** - View information about jobs in scheduling queue
- A few common commands:

-A, --account=<account_list>	Filter by accounts (comma-separated list)
j, --jobs=<job_id_list>	Filter by job IDs (comma-separated list)
-p, --partition=<partition_list>	Filter by partitions (comma-separated list)
-u, --user=<user_list>	Filter by users (comma-separated list)

SLURM Environment Variables

Internal ENV variables that exist when job submitted:

INPUT ENVIRONMENT VARS

- Upon startup, sbatch will read and handle the options set in the following environment variables.
- SBATCH_JOB_NAME
  - Same as -J, --job-name
- SBATCH_ACCOUNT
  - Same as -A, --account
- SBATCH_TIMELIMIT
  - Same as -t, --time
- More…

OUTPUT ENVIRONMENT VARS

- The Slurm controller will set the following variables in the environment of the batch script.
- SLURM_EXPORT_ENV
  - Same as --export.
- SLURM_JOB_ID
  - The ID of the job allocation.
- SLURM_JOB_NAME
  - Name of the job.
- More…

Slurm Partitions on Expanse

Partition Name	Max Walltime	Max Nodes/Job	Max Running Jobs	Max Running + Queued Jobs	Charge Factor	Notes
compute	48 hrs	32	32	64	1	Used for exclusive access to regular compute nodes; *limit applies per group*
shared	48 hrs	1	4096	4096	1	Single-node jobs using fewer than 128 cores
gpu	48 hrs	4	4	8 (32 Tres GPU)	1	Used for exclusive access to the GPU nodes
gpu-shared	48 hrs	1	24	24 (24 Tres GPU)	1	Single-node job using fewer than 4 GPUs
large-shared	48 hrs	1	1	4	1	Single-node jobs using large memory up to 2 TB (minimum memory required 256G)
debug	30 min	2	1	2	1	Priority access to shared nodes set aside for testing of jobs with short walltime and limited resources
gpu-debug	30 min	2	1	2	1	Priority access to gpu-shared nodes set aside for testing of jobs with short walltime and limited resources; *max two gpus per job*
preempt	7 days	32		128	.8	Non-refundable discounted jobs to run on free nodes that can be pre-empted by jobs submitted to any other queue
gpu-preempt	7 days	1		24 (24 Tres GPU)	.8	Non-refundable discounted jobs to run on unallocated nodes that can be pre-empted by higher priority queues

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

Slurm Commands

- sacct
- sacctmgr
- salloc
- sattach
- sbatch
- sbcast
- scancel
- scontrol
- scrontab
- sdiag
- sh5util
- sinfo
- sprio
- squeue
- sreport
- srun
- sshare
- sstat
- strigger
- sview

Common Slurm Command Examples

- Submit jobs using the sbatch command:

  $ ***sbatch*** mycode-slurm.sb

  Submitted batch job 8718049

- Check job status using the squeue command:

  $ ***squeue*** -u $USER

  ```
  JOBID PARTITION     NAME     USER      ST       TIME  NODES  NODELIST(REASON)
  8718049   compute       mycode user   PD       0:00       1              (Priority)
  ```

- Once the job is running, monitor its state:

  $ **squeue** -u $USER

  ```
  JOBID PARTITION     NAME     USER    ST       TIME  NODES  NODELIST(REASON)
  8718049     debug       mycode user   R        0:02      1          expanse-14-01
  ```

- Cancel a running job:

  $ ***scancel*** 8718049

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

SLURM “srun” Command

srun --mpi=pmi2 -n 24 --cpu-bind=rank ./mpi_prime Y

- Used to launch a parallel job on cluster managed by Slurm.
- If necessary, srun will first create a resource allocation in which to run the parallel job.
- Common arguments used on Expanse:
  - --mpi=<*mpi_type*>Identify the type of MPI to be used. Use ‘pmi2’
  - -n, --ntasks=<*number*>Specify the number of tasks to run.
  - -cpu-bind: bind tasks to CPUs
- what is the difference between mpirun and SLURM srun?
  - srun is optimized for Expanse (via the PMI interface) and more efficiently allocates, organizes, and starts up the MPI processes.

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

Outline

- Introduction to Batch Jobs
- What is a Batch Scheduler
- Anatomy of a Batch Script
- Using the SLURM Environment
- Using Batch Scripts on Expanse

General Steps: Compiling/Running Jobs

- Change to your working directory (e.g. hpctr-examples):

```
[mthomas@login02 ~]$
[mthomas@login02 ~]$ cd /home/$USER/hpctr-examples/mpi
[mthomas@login02 mpi]$ ll hello_mpi.f90
-rw-r--r-- 1 mthomas use300 333 Feb 18 13:42 hello_mpi.f90
[mthomas@login02 mpi]$
```

- Verify that the correct modules are loaded:

```
[mthomas@login02 ~]$ module list
Currently Loaded Modules:
  1) shared   2) cpu/0.15.4   3) slurm/expanse/21.08.8
  4) sdsc/1.0   5) DefaultModules
```

- Compile MPI hello world code:

```
[mthomas@login02 mpi]$
[mthomas@login02 mpi]$ mpif90 -o hello_mpi hello_mpi.f90
[mthomas@login02 mpi]$
```

- Verify executable create date:

```
[mthomas@login02 mpi]$
[mthomas@login02 mpi]$ ls -lt hello_mpi
-rwxr-xr-x 1 mthomas use300 22440 Jun 26 18:45 hello_mpi
[mthomas@login02 mpi]$
```

- Submit job

```
[mthomas@login02 mpi]$
[mthomas@login02 mpi]$ sbatch hellompi-slurm.sb
Submitted batch job 14124495
[mthomas@login02 mpi]$
```

MPI Hello World

- Change to the MPI examples directory:

```
[user@login02 MPI]$ cat hello_mpi.f90
!  Fortran example
   program hello
   include 'mpif.h'
   integer rank, size, ierror, tag, status(MPI_STATUS_SIZE)

   call MPI_INIT(ierror)
   call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierror)
   call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierror)
   print*, 'node', rank, ': Hello world!'
   call MPI_FINALIZE(ierror)
   end
[user@login02 MPI]$
```

MPI Hello World: Compile

Set the environment and then compile the code

```
[user@login02 MPI]$ cat README.txt
[1] Compile:

# Load module environment
module purge
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4

mpif90 -o hello_mpi hello_mpi.f90

[2a] Run using Slurm:

sbatch hellompi-slurm.sb

[2b] Run using Interactive CPU Node

srun --partition=debug --account=sds184 --pty --nodes=1 --ntasks-per-
node=128 --mem=248 -t 00:30:00 --wait=0 --export=ALL /bin/bash
```

```
[user@login02 MPI]$ module list

Currently Loaded Modules:
  1) cpu/1.0   2) slurm/expanse/20.02.3

[user@login02 MPI]$ module purge
[user@login02 MPI]$ module load slurm
[user@login02 MPI]$ module load cpu
[user@login02 MPI]$ module load gcc/10.2.0
[user@login02 MPI]$ module load openmpi/4.0.4
[user@login02 MPI]$ module list

Currently Loaded Modules:
  1) slurm/expanse/20.02.3   2) cpu/1.0   3) gcc/10.2.0   4) openmpi/4.0.4

[user@login02 MPI]$ mpif90 -o hello_mpi hello_mpi.f90
[user@login02 MPI]$
```

MPI Hello World: Batch Script

- To run the job, use the batch script submission command.
- Monitor the job until it is finished using the squeue command.

```
[user@login02 MPI]$ cat hellompi-slurm-gnu.sb
#!/bin/bash
#SBATCH --job-name="hellompi-gnu"
#SBATCH --output="hellompi-gnu.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=128
#SBATCH --export=ALL
#SBATCH –A abc123
#SBATCH -t 00:10:00

#This job runs with 2 nodes,
128 cores per node for a total of 256 cores.

## Environment
module purge
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4

## Use srun to run the job

srun --mpi=pmi2 -n 256 --cpu-bind=rank ./hello_mpi_gnu

[user@login02 MPI]$
```

```
[user@login02 MPI]$ sbatch hellompi-slurm-gnu.sb; squeue -u user
Submitted batch job 108910
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            108910   compute hellompi  user PD       0:00      2 (None)
[user@login02 MPI]$ cat hellompi-gnu.108910.exp-12-54.out
 node         4 : Hello world!
 node         5 : Hello world!
 node         7 : Hello world!
 node         0 : Hello world!
 node         2 : Hello world!
 node         3 : Hello world!
 node         9 : Hello world!
 node        10 : Hello world!

[SNIP]

node       247 : Hello world!
 node       248 : Hello world!
 node       249 : Hello world!
 node       186 : Hello world!
 node       220 : Hello world!
 node       203 : Hello world!
 node       135 : Hello world!
```

Passing values into the batch script

- For SLURM: use the **--export** flag.
  - For other schedulers check documentation
- Example: pass the value of two variables x and B into the job script named jobscript.sbatch
  - sbatch --export=x=7,B=‘mystring' jobscript.sb
  - OR: sbatch --export=ALL,x=7,B= mystring ' jobscript.sbatch
- The first example will replace the user's environment with a new environment containing only values for x and B and the SLURM_* environment variables. The second will add the values for A and b to the existing environment.

Example: Passing Vars to Batch Script

```
[uswername@login02 calc-prime]$ cat mpi-prime-slurm.sb
#!/bin/bash
#SBATCH --job-name="mpi_prime"
#SBATCH --output="mpi_prime.%j.%N.out"
####SBATCH --partition=compute
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=24
#SBATCH --export=ALL
#SBATCH -t 00:10:00
#SBATCH -A use300
## Environment
module purge
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4
## echo job name and id:
echo "SLURM_JOB_NAME:  $SLURM_JOB_NAME"
echo "SLURM_JOB_ID:    $SLURM_JOB_ID"
d=`date`
echo "DATE:  $d"
echo "Var NHI:  $NHI"
## Use srun to run the job, pass variable to code
srun --mpi=pmi2 -n 24 --cpu-bind=rank ./mpi_prime $NHI
```

Batch script showing environment, date, and passing variable

```
[mthomas@login02 calc-prime]$ sbatch --export=NHI=15000 mpi-prime-slurm.sb
Submitted batch job 14825136
[mthomas@login02 calc-prime]$ !sq
squeue -u mthomas
JOBID PARTITION     NAME     USER ST   TIME  NODES NODELIST(
14825136     debug mpi_prim  mthomas  R       0:04      1 exp-9-55
[mthomas@login02 calc-prime]$ cat  mpi_prime. 14825136.exp-9-55.out
SLURM_JOB_NAME:  mpi_prime   SLURM_JOB_ID:    14825136
DATE:  Fri Jan 28 14:54:54 PST 2022
Var NHI:  15000
The argument supplied is 250000
PRIME_MPI.     n_hi= 250000     C/MPI version
         N        Pi          Time
         1         0       0.000361
         2         1       0.000004
         4         2       0.000768
         8         4       0.000003
        16         6       0.000003
        32        11       0.000003
        64        18       0.000003
       128        31       0.000004
       256        54       0.000554
[SNIP]
```

Setting Job Priorities

- Jobs are selected to be evaluated by the scheduler in the following order:
  - Jobs that can preempt
  - Jobs with an advanced reservation
  - Partition Priority Tier
  - Job priority
  - Job submit time
  - Job ID
- By default, Slurm assigns job priority on a First In, First Out (FIFO) basis.
- Priority is decided based on a few factors, such as how many jobs the user has run during that month on that cluster, how much time is requested for the job, how many cores and nodes are requested, or how long the job has waited.

- You can find the priority of a job:
  - $ scontrol show job=<job-id>
- Use submission options like:
  - --priority=<value>
  - --nice[=adjustment]

- You can adjust the priority of your own jobs by setting the nice value on their jobs.
  - sbatch --nice=POSITIVE_NUMBER script.sh

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

Checking & Setting Priorities

```
[mthomas@login01 calc-prime]$ scontrol show job=14823516_294
JobId=14824115 ArrayJobId=14823516 ArrayTaskId=294 JobName=CASM2
   UserId=jholber(531677) GroupId=mia326(11764) MCS_label=N/A
   Priority=7176 Nice=0 Account=mia326 QOS=shared-normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:01:25 TimeLimit=12:00:00 TimeMin=N/A
   SubmitTime=2022-08-01T08:11:52 EligibleTime=2022-08-01T08:11:52
   AccrueTime=2022-08-01T08:44:30
   StartTime=2022-08-01T10:26:45 EndTime=2022-08-01T22:26:45 Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2022-08-01T10:26:45
Scheduler=Backfill
   Partition=shared AllocNode:Sid=exp-10-58:2721107
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=exp-13-45
   BatchHost=exp-13-45
   NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=1,mem=1000M,node=1,billing=3600
   Socks/Node=* NtasksPerN:B:S:C=1:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=1000M MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/home/jholber/mechanoChemMl_active_learning/examples/active_learning/exploit_
hessian/submit.slrm
   WorkDir=/home/jholber/mechanoChemMl_active_learning/examples/active_learning/exploit_he
ssian
   StdErr=/home/jholber/mechanoChemMl_active_learning/examples/active_learning/exploit_hess
ian/outputFiles/errors.%J
   StdIn=/dev/null
   StdOut=/home/jholber/mechanoChemMl_active_learning/examples/active_learning/exploit_hes
sian/outputFiles/output.%J
   Power=
```

```
[mthomas@login01 env_info]$ squeue -u mthomas
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
          14825263   compute  envinfo  mthomas PD       0:00      1 (Priority)
[mthomas@login01 env_info]$ sbatch --nice=20 env-slurm.sb
Submitted batch job 14825269
[mthomas@login01 env_info]$ sbatch --priority=1000 env-slurm.sb
Submitted batch job 14825272
[mthomas@login01 env_info]$ squeue -u mthomas
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
          14825272   compute  envinfo  mthomas PD       0:00      1 (Priority)
          14825263   compute  envinfo  mthomas PD       0:00      1 (Priority)
          14825269   compute  envinfo  mthomas PD       0:00      1 (Priority)
```

- Success of these policies affect only your jobs
- ➔ you can move a job up in your queue of jobs, but you will not affect other jobs

Other Batch Topics To Think About

- Using shell script syntax is supported
- Backfill is a scheduling optimization that allows a scheduler to make better use of available resources by running jobs out of order.
- Checkpointing and Restarts: improve user jobs' throughput by exploiting the holes in the Slurm schedule and saves state for when things go wrong.
  - Typically done manually within your code
- Job dependencies: used to defer the start of a job until the specified dependencies have been satisfied
  - SLURM capability exists

Conclusions

- Batch Scripts are used to run jobs on the HPC system in the background
- You can develop scripts with advanced task management (loops), do scaling studies, stage files on and off the compute nodes
- Explore the use of Environment Variables, importing variables to the script, etc.

Resources

- Expanse User Guide & Tutorial
  - https://www.sdsc.edu/support/user_guides/expanse.html
  - https://hpc-training.sdsc.edu/expanse-101/
- SDSC Training Resources
  - https://www.sdsc.edu/education_and_training/training_hpc.html
  - https://github.com/sdsc-hpc-training-org/hpctr-examples
- For this presentation:
  - https://slurm.schedmd.com/tutorials.html
  - https://hpc-wiki.info/hpc/Scheduling_Basics

Thank You