Tufts Cluster Update: Introduction to Slurm

David Lapointe
December 15, 2014
What’s New!

• Increased Hardware
  – Cisco
    • UCS Compute nodes (1000 cores)
    • Dual 10 core/socket nodes
    • Nodes: alpha/omega
  – IBM Hardware
    • Decommissioned old HS21 and HS22 hw (~500 cores)
    • Migrate M3 and M4 compute nodes progressively
    • Oct 15- moved 1/3 of M3/M4 compute nodes.
    • Nov 17 – moved 1/3 M3 nodes
    • Dec 17 – last 1/3 moved

• Slurm Scheduler
  – Currently on UCS cluster
  – Replaces LSF

• Final (12/2014): ~2600 cores, 12 nVidia GPU, Slurm
More Newness

• login.cluster.tufts.edu
  – No file transfers here
• xfer.cluster.tufts.edu
  – Transfer node sftp/scp/rsync
  – No ssh login to xfer
• ssh to nodes
  – Only when node is assigned via slurm
• Private spaces during login
• Modules can be invoked within scripts
Agenda

- What is Slurm?
- Basics of Slurm
- Examples
What is Slurm?
Slurm Facts

• Began at Lawrence Livermore National Laboratory (2003)
• TOP500:
  – 6 of the top 10 sites use Slurm
  – Half of the Top500 use Slurm
• Locally Harvard and MIT have moved to Slurm for their HPC clusters (MGHPCC)
• Developers worldwide contribute to Slurm development
• Open Source /Flexible /Extensible
Slurm Workload Manager

Slurm is an open-source workload manager designed for Linux clusters of all sizes. It provides three key functions. First it allocates exclusive and/or non-exclusive access to resources (computer nodes) to users for some duration of time so they can perform work. Second, it provides a framework for starting, executing, and monitoring work (typically a parallel job) on a set of allocated nodes. Finally, it arbitrates contention for resources by managing a queue of pending work.

Slurm’s design is very modular with dozens of optional plugins. In its simplest configuration, it can be installed and configured in a couple of minutes (see Caos NSA and Perceus: All-in-one Cluster Software Stack by Jeffrey B. Layton). More complex configurations can satisfy the job scheduling needs of world-class computer centers and rely upon a MySQL database for archiving accounting records, managing resource limits by user or bank account, or supporting sophisticated job prioritization algorithms.

While other workload managers do exist, Slurm is unique in several respects:

- **Scalability**: It is designed to operate in a heterogeneous cluster with up to tens of millions of processors.
- **Performance**: It can accept 1,000 job submissions per second and fully execute 500 simple jobs per second (depending upon hardware and system configuration).
- **Free and Open Source**: Its source code is freely available under the GNU General Public License.
- **Portability**: Written in C with a GNU autoconf configuration engine. While initially written for Linux, Slurm has been ported to a diverse assortment of systems.
- **Power Management**: Job can specify their desired CPU frequency and power use by job is recorded. Idle resources can be powered down until needed.
- **Fault Tolerant**: It is highly tolerant of system failures, including failure of the node executing its control functions.
- **Flexibility**: A plugin mechanism exists to support various interconnects, authentication mechanisms, schedulers, etc. These plugins are documented and simple enough for the motivated end user to understand the source and add functionality.
Slurm

- **Simple Linux Utility for Resource Management**
  - Runs on other OSes also

- **Three basic functions**
  - Allocate access to resources (nodes)
  - Provide a framework for managing work
    - Start, execute, monitor jobs
  - Arbitrate contentions for resources by managing pending work requests
Resource Manager

• Allocate Resources
  – Nodes
    • Sockets
      – Cores
        » Hyperthreads
    • Memory
    • GPU

• Launch and Manage Jobs
Scheduler

• When there are more jobs than resources a scheduler is needed to manage queues/partitions

• Complex algorithm for assigning resources
  – Fair-share, reservations, preemption, prioritization
  – Resource limits (queue, user, group)
  – QOS

• Slurm is both a resource manager and a scheduler
“You have to get in line to ride”
Basics of Slurm
Partitions and Jobs
### Slurm <-> LSF

<table>
<thead>
<tr>
<th>Action</th>
<th>Slurm</th>
<th>LSF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show running jobs</td>
<td>squeue</td>
<td>bjobs</td>
</tr>
<tr>
<td>Submit a batch job</td>
<td>sbatch</td>
<td>bsub</td>
</tr>
<tr>
<td>Submit with allocations</td>
<td>salloc</td>
<td>bsub + options + queue</td>
</tr>
<tr>
<td>Start an interactive session</td>
<td>srun</td>
<td>bsub –lp –q int_public6 &lt;app&gt;</td>
</tr>
<tr>
<td>List queues/partitions</td>
<td>squeue</td>
<td>bqueues</td>
</tr>
<tr>
<td>List nodes</td>
<td>sinfo</td>
<td>bhosts</td>
</tr>
<tr>
<td>Control running jobs</td>
<td>scontrol</td>
<td>bstop</td>
</tr>
<tr>
<td>Kill a job</td>
<td>scancel</td>
<td>bkill</td>
</tr>
<tr>
<td>User accounting</td>
<td>sreport or sacct</td>
<td>bacct</td>
</tr>
<tr>
<td>Other functions</td>
<td>srun</td>
<td>lsrun, lsplace</td>
</tr>
</tbody>
</table>
Partitions

• Batch
  – Submit serial programs for execution
  – `sbatch {options} –o outfile mybatch.sh`

• Interactive
  – Run an interactive program
  – Higher priority than batch, same priority as MPI
  – `srun --x11=first --pty --c4 -p interactive matlab`

• MPI (Message Passing Interface)
  – Submit parallel programs for execution
  – `sbatch {options} –p mpi mpi_script.sh`
  – Higher priority than batch, same priority as Interactive

• Interactive and MPI can preempt batch jobs, but do not preempt each other.
### Slurm Partitions

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>AVAIL</th>
<th>TIMELIMIT</th>
<th>NODES</th>
<th>STATE</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpu</td>
<td>up</td>
<td>1-00:00:00</td>
<td>2</td>
<td>idle</td>
<td>alpha025,omega025</td>
</tr>
<tr>
<td>interactive</td>
<td>up</td>
<td>4:00:00</td>
<td>2</td>
<td>idle</td>
<td>alpha001,omega001</td>
</tr>
<tr>
<td>largemem</td>
<td>up</td>
<td>5-00:00:00:00</td>
<td>4</td>
<td>idle</td>
<td>alpha[022-023],omega[022-023]</td>
</tr>
<tr>
<td>batch*</td>
<td>up</td>
<td>3-00:00:00</td>
<td>96</td>
<td>idle</td>
<td>alpha[002-024],omega[002-024]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>m3n[01-50]</td>
</tr>
<tr>
<td>mpi</td>
<td>up</td>
<td>7-00:00:00</td>
<td>46</td>
<td>idle</td>
<td>alpha[002-024],omega[002-024]</td>
</tr>
</tbody>
</table>

The default partition, marked with an asterisk, is batch. Most jobs will be submitted to the batch partition, in the form of a script, as serially executed jobs. (examples follow)

For jobs which use parallel processing, the mpi partition should be used.

The interactive partition is useful for running programs and applications which rely on user interaction, or are graphical based. Development work can be done using the interactive partition.

Applications which use gpus can be run on the gpu partition. There are just 2 GPU nodes in the Cisco system currently, each one with 2 nVidia GPUs.
Slurm Options

• Most options have two formats: short/long
  • The short option uses a single – with a letter
  • The long option uses double - - with a string
  • Eg -p mpi
    --partition mpi

• Time format day-hour:minute:seconds
  • -t 2-10:00:00 for two days and 10 hours
  • -t 00:05:00 for five minutes
  • -t 5 alternate style for 5 minutes
Slurm Options

- **-N** request nodes e.g. –N 5
- **-n** request tasks e.g. –n 4
- **-c** request cores e.g. –c 8
- **--pty** for interactive
- **--x11=first** add X11 to an interactive session
  - Matlab, mathematica, maple, Rstudio, etc
  - Need to login with X11 ( ssh -X or ssh –Y)
- **--mem** allocates total memory
  **--mem=4000**
  ( value in MB )
- **--mem-per-cpu** allocates memory per core
  **-- mem-per-cpu=2000**
Resource Limits

Slurm is strict about resource limits

• JobPerUser 256 max
• CoresPerUser 256 max
• RunTimes - hard limits
  – Batch 3 days max
  – Largemem 5 days max
  – MPI 7 days max
  – Interactive 4 hours max
  – GPU 1 day max
• Memory allocation is a hard limit
  – 2 Gb per core is default, can be increased
  – Understand your memory requirements
Examples
sinfo shows status of the partitions.

```
$sinfo

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>AVAIL</th>
<th>TIMELIMIT</th>
<th>NODES</th>
<th>STATE</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpu</td>
<td>up</td>
<td>3-00:00:00</td>
<td>2</td>
<td>idle</td>
<td>alpha025,omega025</td>
</tr>
<tr>
<td>largemem</td>
<td>up</td>
<td>5-00:00:00</td>
<td>1</td>
<td>mix</td>
<td>omega023</td>
</tr>
<tr>
<td>largemem</td>
<td>up</td>
<td>5-00:00:00</td>
<td>3</td>
<td>idle</td>
<td>alpha[022-023],omega022</td>
</tr>
<tr>
<td>interactive</td>
<td>up</td>
<td>4:00:00</td>
<td>1</td>
<td>mix</td>
<td>alpha001</td>
</tr>
<tr>
<td>interactive</td>
<td>up</td>
<td>4:00:00</td>
<td>1</td>
<td>idle</td>
<td>omega001</td>
</tr>
<tr>
<td>batch*</td>
<td>up</td>
<td>3-00:00:00</td>
<td>14</td>
<td>down*</td>
<td>m3n[37-50]</td>
</tr>
<tr>
<td>batch*</td>
<td>up</td>
<td>3-00:00:00</td>
<td>22</td>
<td>mix</td>
<td>alpha[002-009,024],m3n23,omega[002-012,023]</td>
</tr>
<tr>
<td>batch*</td>
<td>up</td>
<td>3-00:00:00</td>
<td>36</td>
<td>alloc</td>
<td>m3n[01-22,24-36],omega024</td>
</tr>
<tr>
<td>batch*</td>
<td>up</td>
<td>3-00:00:00</td>
<td>24</td>
<td>idle</td>
<td>alpha[010-023],omega[013-022]</td>
</tr>
<tr>
<td>mpi</td>
<td>up</td>
<td>7-00:00:00</td>
<td>21</td>
<td>mix</td>
<td>alpha[002-009,024],omega[002-012,023]</td>
</tr>
<tr>
<td>mpi</td>
<td>up</td>
<td>7-00:00:00</td>
<td>1</td>
<td>alloc</td>
<td>omega024</td>
</tr>
<tr>
<td>mpi</td>
<td>up</td>
<td>7-00:00:00</td>
<td>24</td>
<td>idle</td>
<td>alpha[010-023],omega[013-022]</td>
</tr>
<tr>
<td>m4</td>
<td>up</td>
<td>7-00:00:00</td>
<td>40</td>
<td>down*</td>
<td>m4c[21-60]</td>
</tr>
<tr>
<td>m4</td>
<td>up</td>
<td>7-00:00:00</td>
<td>18</td>
<td>alloc</td>
<td>m4c[03-20]</td>
</tr>
<tr>
<td>m4</td>
<td>up</td>
<td>7-00:00:00</td>
<td>2</td>
<td>idle</td>
<td>m4c[01-02]</td>
</tr>
</tbody>
</table>
```
**SQUEUE**

`squeue` show the status of jobs on the cluster

```
$squeue
```

**Example**

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>Nodelist(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5866951</td>
<td>batch</td>
<td>20_traff</td>
<td>hcui01</td>
<td>R</td>
<td>21:12:10</td>
<td>1</td>
<td>m3n04</td>
</tr>
<tr>
<td>5866952</td>
<td>batch</td>
<td>20_traff</td>
<td>hcui01</td>
<td>R</td>
<td>21:12:10</td>
<td>1</td>
<td>m3n05</td>
</tr>
<tr>
<td>5880203</td>
<td>interacti</td>
<td>pseq</td>
<td>ewagne05</td>
<td>R</td>
<td>9:21</td>
<td>1</td>
<td>alpha001</td>
</tr>
<tr>
<td>5880198</td>
<td>largemem</td>
<td>blast2hi</td>
<td>cburke04</td>
<td>R</td>
<td>24:36</td>
<td>1</td>
<td>omega023</td>
</tr>
<tr>
<td>5835669</td>
<td>largemem</td>
<td>sbatch</td>
<td>shiger01</td>
<td>R</td>
<td>2-18:37:31</td>
<td>1</td>
<td>omega023</td>
</tr>
<tr>
<td>5880204</td>
<td>mpi</td>
<td>1.83</td>
<td>hyu04</td>
<td>R</td>
<td>0:06</td>
<td>1</td>
<td>omega013</td>
</tr>
<tr>
<td>5838915</td>
<td>mpi</td>
<td>AA21</td>
<td>smchug04</td>
<td>R</td>
<td>1-12:08:54</td>
<td>8</td>
<td>alpha[002-009]</td>
</tr>
<tr>
<td>5835764</td>
<td>mpi</td>
<td>test</td>
<td>smchug04</td>
<td>R</td>
<td>1-21:14:44</td>
<td>8</td>
<td>omega[005-012]</td>
</tr>
</tbody>
</table>

```
$squeue –help
```

Displays options for `squeue`

```
$squeue –u my_utln
```

Displays job status for user `my_utln`
JOB STATE CODES

Jobs typically pass through several states in the course of their execution. The typical states are PENDING, RUNNING, SUSPENDED, COMPLETING, and COMPLETED. An explanation of each state follows.

CD COMPLETED  Job has terminated all processes on all nodes.

CG COMPLETING Job is in the process of completing. Some processes on some nodes may still be active.

F FAILED  Job terminated with non-zero exit code or other failure condition.

NF NODE_FAIL  Job terminated due to failure of one or more allocated nodes.

PD PENDING  Job is awaiting resource allocation.

PR PREEMPTED  Job terminated due to preemption.

R RUNNING  Job currently has an allocation.

S SUSPENDED  Job has an allocation, but execution has been suspended.

TO TIMEOUT  Job terminated upon reaching its time limit.
SCANCEL Example

SCANCEL allows jobs to be cancelled.

```
$ sbatch -n2 test_job
Submitted batch job 259

$ squeue

    JOBID PARTITION     NAME     USER ST   TIME  NODES NODELIST(REASON)
    259    batch test_job dlapoi01  R       0:10      1 alpha003

$ scancel 259

Or

$ scancel -signal=TERM 259
```
Shell Directives

Shell directives are Slurm options that can be specified in a script, similar to the command line options.

Shell directives start with a #SBATCH followed by options and values

Useful options set memory, time, and other resource allocations

--mem=500          MB allocated for run
--mem-per-cpu=250   MB allocated per core
-t 5              Length of time (minutes) allocated to the job
-N 4              Number of Nodes requested for job
-c 16             Number of cores requested

Example if –N 1 –n 2 are specified both tasks will be on the same node.
--mem=4000    4Gb
Or
--mem-per-cpu=4000  -c2 each core will get 4 Gb for a total of 8 Gb.

To see a complete list of directives, visit the sites on the Resources Slide, or use the man pages for sbatch, salloc, srun.
Using SBATCH to run scripts

Submitting scripts using `sbatch` is the main way to run jobs on the cluster.

`sbatch myjobscript.sh`

All options can be listed in the script, for example

```bash
#!/bin/bash
#SBATCH --c 2         # request two cores
#SBATCH --p batch     # partition to run job on
#SBATCH --mem=1000    # total memory (MB) request for job, split over cores
#SBATCH --o outputfile.out
#SBATCH --e error.out
#SBATCH --mail-type=END # notification email sent when job finishes (END)
#SBATCH --mail-user=my_utln@tufts.edu

module load  ##### # modules needed for script

{script....}
```
SBATCH Example

```
$ cat test_job
#!/bin/bash
echo "I'm on the launching node!"
hostname
echo "These tasks are on the allocated nodes"
srun -l hostname
srun -l sleep 10
echo "All done"

$ sbatch -n2 test_job
Submitted batch job 259

$ squeue
    JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
        259    batch test_job dlapoi01  R       0:10      1 alpha003

$ cat slurm-259.out
I'm on the launching node!
alpha003
These tasks are on the allocated nodes
 0: alpha003
 1: alpha003
All done
```
SBATCH: Wrap

Perhaps you have something like this.

```
prog1 params|prog2 - params2|prog3 >output
```

Could write a batch script for this, but there is an easier option

```
--wrap=<command string>
```

`sbatch` will wrap the specified command string in a simple "sh" shell script, and submit that script to the slurm controller. When `--wrap` is used, a script name and arguments may not be specified on the command line; instead the sbatch-generated wrapper script is used.

Might want to `salloc` resources first.

```
sbatch --wrap="prog1 params|prog2 - params2|prog3 >output"
```

**Don’t do this with `srun`: The pipes won’t work.**

```
srun prog1 params|prog2 - params2|prog3 >output
```
#!/bin/bash
## runBWA.sh script

#SBATCH -c 10
#SBATCH --mem 16000
#SBATCH -p batch
#SBATCH -o runBWA.out
#SBATCH -e runBWA.err
#SBATCH --mail-type=END
#SBATCH --mail-user=david.lapointe@tufts.edu

module load bwa/0.7.9a
module load samtools/0.1.19

fnames=('DN3_ROI/DN3' 'HH14_ROI/HH14')

export MM10=/cluster/tufts/genomes/MusMusculus/mm10/Sequence/BWAIndex

for i in "${fnames[@]}"; do
  SORTED=$i"_sorted"
  bwa mem -t 10 $MM10/genome.fa $i.fasta > $i.sam
  samtools view -b -S $i.sam >$i.bam
  samtools sort $i.bam $SORTED
  samtools index $SORTED.bam $SORTED.bai
done

$sbatch runBWA.sh   # will run one set then the other
srun allows running a job on the a specified partition. Unless specified the default partition is used. srun can be used within salloc, as part of a script sent to sbatch, or by itself for an interactive session.

```
$srun -l -N3 hostname
1: alpha004
0: alpha005
2: alpha003
```

srun is useful for running interactive jobs, where applications are graphically based, or console tools, like R

```
$salloc --mem=8000 --c10
salloc: Granting job allocation 258
srun --x11=first --pty --p interactive matlab
```

Or

```
$srun --mem=8000 --c10 --x11=first --pty --p interactive matlab
```
SALLOC allocates resources

`salloc` is used to allocate resources which can be used repeatedly until exit which then relinquishes the resources (Nodes, cores, memory,...).

Example

```
$ salloc -N5 sh  # allocate 5 nodes and start a shell locally
salloc: Granted job allocation 255
sh-4.1$ srun hostname
alpha003
alpha004
alpha005
alpha006
alpha007
sh-4.1$ exit
exit
salloc: Relinquishing job allocation 255
```

Be sure to release the allocations using `exit` when you are finished.

```
$ salloc -N 4 mpirun -nolocal mrbayes-3.2_p.mb test.txt
```

can be used to run mpi jobs.
**MPI**

*(Message Passing Interface)*

MPI runs should use the MPI partition, with `salloc`, `sbatch`, or both. MPI does not work well using `srun`.

```bash
$ salloc -N 4 -p mpi sh  # allocates four nodes, mpi and a local shell
salloc: Granted job allocation 562
sh4.1$ module load openmpi/1.8.2
sh4.1$ mpirun hello_world
Hello world from processor alpha003, rank 1 out of 4 processors
Hello world from processor alpha005, rank 3 out of 4 processors
Hello world from processor alpha002, rank 0 out of 4 processors
Hello world from processor alpha004, rank 2 out of 4 processors
sh4.1$ exit
exit
salloc: Relinquishing job allocation 562
salloc: Job allocation 562 has been revoked.
```
MPI
(Message Passing Interface)

MPI runs should use the MPI partition, with salloc, sbatch, or both. Load the openmpi/1.8.2 module first!

$module load openmpi/1.8.2
$salloc -N 4 -p mpi mpirun hello_world
salloc: Granted job allocation 560
Hello world from processor alpha003, rank 1 out of 4 processors
Hello world from processor alpha005, rank 3 out of 4 processors
Hello world from processor alpha002, rank 0 out of 4 processors
Hello world from processor alpha004, rank 2 out of 4 processors
salloc: Relinquishing job allocation 560
salloc: Job allocation 560 has been revoked.
MPI
(Message Passing Interface)

MPI runs should use the MPI partition, with salloc, sbatch, or both.

$ sbatch -N 4 -p mpi hello_test.sh
Submitted batch job 556
[dlapoi01@login001 mpi_hello_world]$ more slurm-556.out
Hello world from processor alpha002, rank 0 out of 4 processors
Hello world from processor alpha004, rank 2 out of 4 processors
Hello world from processor alpha003, rank 1 out of 4 processors
Hello world from processor alpha005, rank 3 out of 4 processors
Job Arrays

- Job arrays offer a mechanism for submitting and managing collections of similar jobs quickly and easily.
- All jobs must have the same initial options (e.g. size, time limit, etc.).
- These can be altered after the job has begun.
- Job arrays are limited to batch job submissions.
- This groups jobs together but as one submission (job id).
Job Array Example

```bash
$ sbatch --array=1-31 arrtempl
# submits an array of 31 jobs to one node

#!/bin/bash
#SBATCH --j arrtempl      # job name
#SBATCH --o arrtempl_%A_%a.out
#SBATCH --e arrtempl_%A_%a.err
#SBATCH --N 1
#SBATCH --p batch
#SBATCH --t 4:00:00       # four hours run time
#SBATCH --mem 4000

arrprog data$SLURM_ARRAY_TASK_ID.dat

This will submit 31 jobs that use data1.dat .. data31.dat as input data, and have 31 output and error files as arrtempl_jobid_{1..31}.{out,err}

%A is the job id
%a is the array job index (1-31)
$SLURM_ARRAY_TASK_ID is an environment variable which contains the job array index and can be used in the script.
```
Job Dependencies

Under some circumstances jobs submitted serially need to wait for a previous job to finish, for example in a workflow pipeline.

The \texttt{--dependency} option can be used to manage these cases.

\begin{verbatim}
$ sbatch maptogenome.sh
Submitted batch job 978354
$ sbatch \texttt{--dependency=afterok:978354} annotate_hits.sh
Submitted batch job 978357
\end{verbatim}

The \texttt{annotate_hits.sh} is queued but won’t run unless \texttt{maptogenome.sh} is successful (afterok:978354).

There are other parameters that can be used with \texttt{--dependency}, and multiple jobids can be specified. See \texttt{man sbatch} then look for \texttt{--dependency}.
Resources

• go.tufts.edu/cluster - documentation and examples for Tufts HPC
  – Look under slurm for
    • This presentation
    • Command equivalents (Slurm<-> LSF options )

• slurm.schedmd.com - Main site
  – Videos, tutorials, FAQ, documentation

• Man pages on cluster
  – man {slurm|srun|sbatch|salloc|..}
  – man –k slurm  #list of all man pages for slurm
Enj oy
Slurm
IT'S HIGHLY ADDICTIVE!