Introduction to SLURM

for users

Jordi Blasco
(jordi.blasco@nesi.org.nz)

NeSI
New Zealand eScience Infrastructure

Outline

1. Introduction
   - About SLURM
   - New Features introduced with Slurm
   - Resource Management
2. Running a job
   - Job/step allocation
   - Examples - Serial
   - Examples - OpenMP
   - Examples - MPI
   - Examples - Hybrid(MPI+OpenMP)
   - Examples - Array
3. System Information
   - Job monitoring

About SLURM

About SLURM

About SLURM

About SLURM

- **SLURM** was an acronym for Simple Linux Utility for Resource Management.
- Development started in 2002 at Lawrence Livermore National Laboratory as a simple resource manager for Linux clusters
- Has evolved into a capable job scheduler
- About 500,000 lines of C code. Not Simple anymore.
- Now is called : Slurm Workload Manager
- Supports AIX, Linux, Solaris, other Unix variants
- Used on many of the world's largest computers
- Also used in the coolest facility in the world : NeSI
- Commercial support provided by SchedMD

New Features introduced with Slurm

Already deployed features

- Full control over CPU and Memory usage
- Better Scheduling techniques & performance
- Job Array support
- Better integration with MPI
- Interactive sessions support
- High Availability (2 masters)
- Debugger friendly
- Topology aware (better MPI performance)
- Privacy Environment

New Features introduced with Slurm

Future features planed to be deployed

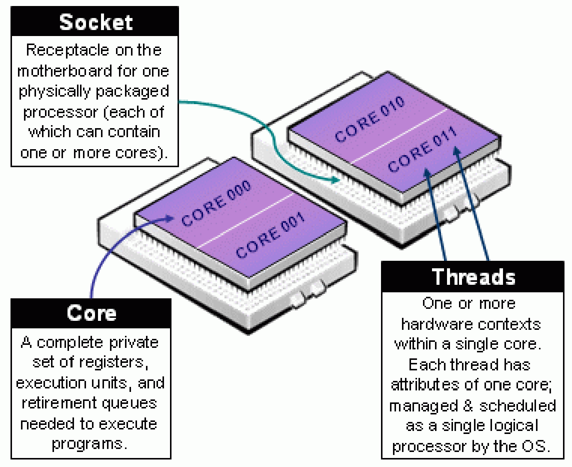
- Kernel Level Checkpointing & Restart
- Job Migration
- Shared FlexLM integration
- Job profiling (`srun --profile=All`)

Resource Management

Node and Job States

- Nodes
  - state (up/down/idle/allocated/mix/drained)
- Jobs
  - queued/pending and running
  - suspended/preempted
  - cancelled/completed/failed

Definitions of Socket, Core, & Thread



Socket

Receptacle on the motherboard for one physically packaged processor (each of which can contain one or more cores).

Core

A complete private set of registers, execution units, and retirement queues needed to execute programs.

Threads

One or more hardware contexts within a single core. Each thread has attributes of one core; managed & scheduled as a single logical processor by the OS.

Figure : Definitions of Socket, Core, & Thread. Source SchedMD

Running a job

SLURM Commands

- sbatch – submits a script job. (=llsubmit)
- scancel – cancels a running or pending job. (=llcancel)
- srun – runs a command across nodes.
- sbcast – Transfer file to a compute nodes allocated to a job.
- interactive – opens an interactive job session.
- sattach – Connect stdin/out/err for an existing job or job step.

Running a job

srun : Simple way to manage MPI, OpenMP, pthreads & serial jobs

- Slurm provides a single command line to manage all the MPI flavours, OpenMP, Pthreads and serial applications
- Users don't need to worry about MPI flags and options for each MPI implementation mpirun/mpiexec/mpiexec.hydra
- The tool is called **srun** and it is **mandatory** for submitting jobs in the cluster.

Running a job

Commonly used SLURM variables

- $SLURM_JOBID
- $SLURM_JOB_NODELIST : (example sb[004,006])
- $SLURM_NNODES (Number of nodes)
- $SLURM_SUBMIT_DIR (Directory from which the job was submitted)

Examples

sbatch

```
login-01.uoa.nesi.org.nz ~ $ vim testjob.sl
login-01.uoa.nesi.org.nz ~ $ cat testjob.sl
#!/bin/bash
#SBATCH --nodes=10
srun echo "running on : $(hostname)"
srun echo "allocation : $SLURM_NODELIST"

login-01.uoa.nesi.org.nz ~ $ sbatch testjob.sl
Submitted batch job 11109

login-01.uoa.nesi.org.nz ~ $ cat slurm-11109.out
running on : wm001
allocation : wm[001-010]
```

Examples

sbatch

```
login-01.uoa.nesi.org.nz ~ $ vim testjob.sl
login-01.uoa.nesi.org.nz ~ $ cat testjob.sl
#!/bin/bash
#SBATCH --nodes=10
srun echo "running on : $(hostname)"
srun echo "allocation : $SLURM_NODELIST"

login-01.uoa.nesi.org.nz ~ $ sbatch testjob.sl
Submitted batch job 11109


login-01.uoa.nesi.org.nz ~ $ cat slurm-11109.out
running on : wm001
allocation : wm[001-010]
```

Examples

sbatch

```
login-01.uoa.nesi.org.nz ~ $ vim testjob.sl
login-01.uoa.nesi.org.nz ~ $ cat testjob.sl
#!/bin/bash
#SBATCH --nodes=10
srun echo "running on : $(hostname)"
srun echo "allocation : $SLURM_NODELIST"

login-01.uoa.nesi.org.nz ~ $ sbatch testjob.sl
Submitted batch job 11109

login-01.uoa.nesi.org.nz ~ $ cat slurm-11109.out
running on : wm001
allocation : wm[001-010]
```

Submitting a Job

Standard Job Script Directives

```
#!/bin/bash
#SBATCH -J JobName
#SBATCH -A uoa99999               # Project Account
#SBATCH --time=08:00:00           # Walltime
#SBATCH --mem-per-cpu=4096        # memory/cpu (in MB)
#SBATCH --ntasks=2                # 2 tasks
#SBATCH --cpus-per-task=4         # number of cores per tasks
#SBATCH --nodes=1                 # number nodes
#SBATCH -C sb                     # sb=Sandybridge,wm=Westmere
```

Submitting a Job

Optional Job Script Directives

```
#SBATCH --mail-type=end
#SBATCH --mail-user=jordi.blasco@nesi.org.nz
#SBATCH -D /path_to_working_directory/
```

Submitting a Serial Job

Job Description Example : Serial

```
#!/bin/bash
#SBATCH -J Serial_JOB
#SBATCH -A uoa99999           # Project Account
#SBATCH --time=01:00:00       # Walltime
#SBATCH --mem-per-cpu=8132  # memory/core (in MB)

srun my_serial_binary
```

Submitting a OpenMP Job

Job Description Example : SMP

```
#!/bin/bash
#SBATCH -J OpenMP_JOB
#SBATCH -A uoa99999          # Project Account
#SBATCH --time=01:00:00      # Walltime
#SBATCH --mem-per-cpu=8132  # memory/core (in MB)
#SBATCH --cpus-per-task=8   # 8 OpenMP Threads

srun my_openmp_binary
```

Submitting a MPI Job

Job Description Example : MPI

```
#!/bin/bash
#SBATCH -J MPI_JOB
#SBATCH -A uoa99999          # Project Account
#SBATCH --time=01:00:00      # Walltime
#SBATCH --mem-per-cpu=8132   # memory/core (in MB)
#SBATCH --ntasks=2           # number of tasks

srun my_mpi_binary
```

Submitting a Hybrid(MPI+OpenMP) Job

Job Description Example : Hybrid(MPI+OpenMP)

```
#!/bin/bash
#SBATCH -J Hybrid_JOB
#SBATCH -A uoa99999           # Project Account
#SBATCH --time=01:00:00       # Walltime
#SBATCH --mem-per-cpu=8132    # memory/core (in MB)
#SBATCH --ntasks=4            # number of tasks
#SBATCH --cpus-per-task=8     # 8 OpenMP Threads
#SBATCH --nodes=1             # Can be range eg --nodes=2-4

srun my_binary_hybrid
```

Submitting an Array Job

Array Job

- Slurm job arrays offer a mechanism for submitting and managing collections of similar jobs quickly and easily.
- In general, array jobs are useful for applying the same processing routine to a collection of multiple input data files.
- Array jobs offer a very simple way to submit a large number of independent processing jobs.

Submitting an Array Job

Array Job Syntax

- Job array with index values between 1 and 1000
  `--array=1-1000`
- Job array with index values of 1, 3, 5 and 7
  `--array=1,3,5,7`
- Job array with index values between 1 and 7 with a step size of 2 (i.e. 1, 3, 5 and 7)
  `--array=1-7:2`

Submitting an Array Job

Array Job example

```
#!/bin/bash
#SBATCH -J JobArray
#SBATCH --time=01:00:00      # Walltime
#SBATCH -A uoa99999          # Project Account
#SBATCH --mem-per-cpu=8132   # memory/core (in MB)
#SBATCH --cpus-per-task=4    # 4 OpenMP Threads
#SBATCH --array=1-1000       # Array definition

srun my_binary_array $SLURM_ARRAY_TASK_ID
```

Submitting an Array Job

Array Job

To submit 1,000 element job array `sbatch blast_array.sl`

Submit time < 1 second

Environment variable with array index: `SLURM_ARRAY_TASK_ID`

Submitting an Array Job

Array Job

The management is really easy:

```
$ squeue -u sbae335
          JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST
  28317_[1-1000]     high  SungSHM  sbae335 PD       0:00      1 (Priority)
27817_[196-1000]     high Sung_BLA  sbae335 PD       0:00      1 (Resources)
      27817_184      high Sung_BLA  sbae335  R    4:15:27      1 wm001
      27817_185      high Sung_BLA  sbae335  R    4:15:27      1 wm001
      27817_186      high Sung_BLA  sbae335  R    4:15:27      1 wm001
      ...
      ...

$ scancel 28317_[900-1000]
```

GRES subsystem

Generic Resource System to request special hardware like GPUs or Intel Phis

Requesting GPUs

Add the following line in your submit script:

```
#SBATCH --gres=gpu:1      # GPUs per node
```

Fine tuning

```
#SBATCH -C kepler # ask only for NVIDIA K20X
#SBATCH -C fermi  # ask only for NVIDIA Tesla M2090
```

GRES subsystem

Requesting GPUs

```
#!/bin/bash
#SBATCH -J GPU_JOB
#SBATCH --time=01:00:00     # Walltime
#SBATCH -A uoa99999         # Project Account
#SBATCH --ntasks=4          # number of tasks
#SBATCH --ntasks-per-node=2 # number of tasks per node
#SBATCH --mem-per-cpu=8132  # memory/core (in MB)
#SBATCH --cpus-per-task=4   # 4 OpenMP Threads
#SBATCH --array=1-1000      # Array definition
#SBATCH --gres=gpu:2        # GPUs per node
#SBATCH -C kepler

srun my_binary_cuda_mpi
```

GRES subsystem

Requesting Intel Phi (MIC)

Add the following line in your submit script:

```
#SBATCH --gres=mic:1   # Intel Phi per node
```

Job dependencies

Job dependencies

Add the following line in your submit script:

```
--dependency=afterok:$SLURM_JOB_ID
```

Running a job

Interactive Job Session

```
[4845] login-01.uoa.nesi.org.nz ~ $interactive -h
Usage: interactive [-A] [-a] [-c] [-m] [-J]

Mandatory arguments:
 -A: account
Optional arguments:
 -a: architecture (default: wm, values sb=SandyBridge wm=Westmere)
 -c: number of CPU cores (default: 1)
 -m: amount of memory (GB) per core (default: 1 [GB])
 -J: job name
example : interactive -A nesi99999 -a wm -c 4 -J MyInteractiveJob

Written by: Alan Orth <a.orth@cgiar.org>
Modified by: Jordi Blasco <jordi.blasco@nesi.org.nz>
```

Limits

Current limits in the cluster

- Max array size : 1000
- Max number of submitted jobs : 10,000
- Max allocatable memory per node : 92GB (Westmere), 124GB (SandyBridge), 508GB LargeMemory nodes)
- Number of cores per node : 12 (Westmere), 16 (SandyBridge), 40 LargeMemory nodes)

Temporary File Systems

Temporary File Systems

- $TMP_DIR (local filesystem)
- $SCRATCH_DIR (shared filesystem)
- $SHM_DIR (local RAM filesystem)

System Information

System Information

- squeue – shows the status of jobs. (=llq)
- sinfo – provides information on partitions and nodes. (=llstatus)
- sview – GUI to view job, node and partition information.
- smap – CLI to view job, node and partition information.

System Information

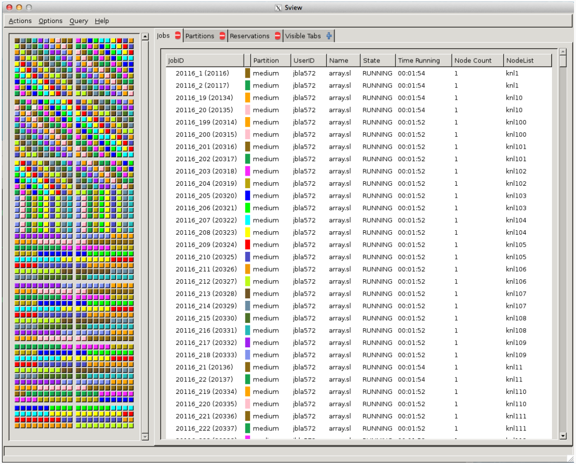
squeue

Show jobid and allocated nodes for running jobs of the user jblasco:

```
[4925] login-01.uoa.nesi.org.nz ~$ squeue
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
24258      high Migrate-  jbla572 PD       0:00      4 (Resources)
24259      high Migrate-  jbla572 PD       0:00      4 (Priority)
24257      high Migrate-  jbla572  R       0:27    512 sb[1-512]
```

System Information

sview



JobID	Partition	UserID	Name	State	Time Running	Node Count	NodeList
20116_1 (20116)	medium	jbla572	array.sl	RUNNING	00:01:54	1	knl1
20116_2 (20117)	medium	jbla572	array.sl	RUNNING	00:01:54	1	knl1
20116_19 (20134)	medium	jbla572	array.sl	RUNNING	00:01:54	1	knl10
20116_20 (20135)	medium	jbla572	array.sl	RUNNING	00:01:54	1	knl10
20116_199 (20314)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl100
20116_200 (20315)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl100
20116_201 (20316)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl101
20116_202 (20317)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl101
20116_203 (20318)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl102
20116_204 (20319)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl102
20116_205 (20320)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl103
20116_206 (20321)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl103
20116_207 (20322)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl104
20116_208 (20323)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl104
20116_209 (20324)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl105
20116_210 (20325)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl105
20116_211 (20326)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl106
20116_212 (20327)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl106
20116_213 (20328)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl107
20116_214 (20329)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl107
20116_215 (20330)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl108
20116_216 (20331)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl108
20116_217 (20332)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl109
20116_218 (20333)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl109
20116_21 (20136)	medium	jbla572	array.sl	RUNNING	00:01:54	1	knl11
20116_22 (20137)	medium	jbla572	array.sl	RUNNING	00:01:54	1	knl11
20116_219 (20334)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl110
20116_220 (20335)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl110
20116_221 (20336)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl111
20116_222 (20337)	medium	jbla572	array.sl	RUNNING	00:01:52	1	knl111

Enjoy
Slurm
Slurm
IT'S HIGHLY ADDICTIVE!!!

Questions & Answers

NeSI

New Zealand eScience Infrastructure