DESY HPC User Meeting – Job Scheduler

> Content

- Role of a job scheduler
- Introduction to SLURM
- SLURM partitions on Maxwell
- Short overview of SLURM configuration
- Best practices

Role of a Job Scheduler

- Usually there is more demand than resources
  - Need to share resources
- Job scheduler manages queue(s) of jobs supporting complex scheduling algorithms
  - Provides fair resource sharing
  - Supports resource limits (user, group, etc.)
  - Node weights
  - Supports reservations
- Provides most efficient usage of the resources
- Calendar based reservation system was

not deployed - no need!

Introduction to SLURM

- Simple Linux Utility for Resource Management
  - Used by many supercomputer centers (including top ones)
  - Manages resources on the compute nodes
  - Schedules jobs using those resources
  - Free and open source (GPL license, active world-wide development, plugins)
  - Fast (1000 job submissions per second)
  - Fault-tolerant
  - Various prolog/epilog scripts
  - We've recently subscribed for a support !

Introduction to SLURM

- Resources in SLURM
  - Nodes – description of a compute node (number of CPUs, memory, etc.)
  - Partitions – group of nodes with specified properties/restrictions
- Basic commands
  - **sinfo** - information about nodes and partitions
  - **squeue** – shows current job queue
  - **sbatch** – submits a job in a batch mode
  - **salloc** – request resources for an interactive job

- More info
  - http://slurm.schedmd.com/
  - https://confluence.desy.de/display/IS/Using+the+Maxwell+Cluster

SLURM Partitions

- Job must be submitted to one of the SLURM partitions
- Partition *maxwell (default)*
  - Includes IT group nodes
  - Every registered user can use it
- Group can have “their” partition
  - Contains compute nodes assigned to the group
  - Only users of a group can submit job (no need to get registered as Maxwell user)
  - Has higher priority than common partition
- Common partition *all*
  - Includes all compute nodes
  - Every user can submit a job
  - Jobs from common partition will be preempted (killed) if running on a nodes of other groups and requeued again
  - Supposed to be used for short(er) or clever(er) jobs

Short overview of SLURM configuration

$ cat /etc/slurm/slurm.conf
$ scontrol show conf (parameters only, including default)

- Fair sharing

```
PriorityType=priority/multifactor
PriorityDecayHalfLife=14-0
PriorityFavorSmall=NO
PriorityMaxAge=7-0
PriorityWeightAge=10000
PriorityWeightFairshare=5000
PriorityWeightJobSize=1000
```

- Some important (more or less) parameters

```
InactiveLimit=3600
SchedulerType=sched/backfill
SelectType=select/linear
```

Short overview of SLURM configuration

$ cat /etc/slurm/slurm.conf

> Node configuration

```
NodeName=max-wn[001-008] Weight=5 RealMemory=256 Sockets=2 CoresPerSocket=8
ThreadsPerCore=2 State=Unknown Feature=INTEL,V3,E5-2640

NodeName=max-wng[003-003] Weight=25 RealMemory=512 Sockets=2
CoresPerSocket=16 ThreadsPerCore=2 State=Unknown
Feature=GPU,K40X,INTEL,V4,E5-2698
```

> Partition configuration

```
PartitionName=DEFAULT GraceTime=300 DefaultTime=0-1:0 State=UP Shared=No
PriorityTier=50 PreemptMode=off

PartitionName=all PriorityTier=10 PreemptMode=requeue Default=NO
MaxTime=14-0:0  MaxNodes=32  AllowGroups=all   Nodes=…

PartitionName=p4 Default=NO MaxTime=14-0:0 MaxNodes=2 AllowGroups=p4_sim
Nodes=max-p4-[001-002]
```

Best practices

- Batch job

```
$ sbatch my_job.sh
```

- MPI program, start 128 processes

my_job.sh

```
#!/bin/bash

#SBATCH --ntasks=128
#SBATCH --partition=all
#SBATCH –time=00:01:00

module load mpi/openmpi-x86_6
mpirun hostname
```

  - Job output to <jobid>.out

Best practices

- Batch job

```
$ sbatch my_job.sh
```

- MPI program, start 32 processes, process needs more memory

my_job.sh

```
#!/bin/bash

#SBATCH --ntasks=32
#SBATCH --cpus-per-task=4
#SBATCH --partition=all
#SBATCH –time=00:01:00

module load mpi/openmpi-x86_6
mpirun hostname
```

Best practices

- Batch job

```
$ sbatch my_job.sh
```

- Multi-thread (OpenMP, …) program, allocate single node

my_job.sh

```
#!/bin/bash

#SBATCH --nodes=1
#SBATCH --partition=all
#SBATCH –time=00:01:00

./a.out
```

Best practices

- Batch job

```
$ sbatch my_job.sh
```

- Hybrid OpenMP/MPI program, allocate 2 nodes, each node starts one MPI process which can then use all cores for multi-threading

```
#!/bin/bash

#SBATCH --ntasks=2
#SBATCH --nodes=2
#SBATCH --partition=all
#SBATCH –time=00:01:00

module load mpi/openmpi-x86_6
mpirun --bind-to none ./a.out       # carefully here
```

Best practices

- Other useful parameters

```
#!/bin/bash

#SBATCH --constraint="INTEL&GPU"  # allocate nodes with specific feature
#SBATCH --no-requeue # disallow requeuing after preemption
#SBATCH --workdir <working directory>
#SBATCH --reservation <reservation name> # use specific reservation

# change output
#SBATCH --output <filename>
#SBATCH --error <filename>

#send e-mails
--mail-type=END,FAIL
--mail-user=<address to send to>
```

- See sbatch docs for more…