Slurm: A quick start tutorial

Slurm is a resource manager and job scheduler. Users can submit jobs (i.e. scripts containing execution instructions) to slurm so that it can schedule their execution and allocate the appropriate resources (CPU, RAM, etc..) on the basis of a user's preferences or the limits imposed by the system administrators. Clearly, the advantages of using slurm on a computational cluster are multiple. For an overview of them please read these pages.

Slurm is **free** software distributed under the GNU General Public License.

What is a parallel job?

A parallel job consists of tasks that run simultaneously. Parallelization can be achieved in different ways, among which:

- by running a multi-process program, for example using OpenMPI.
- by running a multi-threaded program, for example see pthreads.

A multi-process program consists of multiple tasks orchestrated by MPI and possibly executed by different nodes. On the other hand, a multi-threaded program consists of multiple task using several CPUs on the same node.

Slurm's command `srun' (see below) allows users to create tasks and/or request CPUs for a particular task such that both types of parallelizations mentioned above can be achieved easily. For instance, the *–ntasks n (-N)* option will create **n processes**, while the *–cpus-per-task n (-c)* option will created a single **n-threaded process**. Tasks cannot be split across several compute nodes. See the examples below.

Slurm's architecture

Slurm is made of a slurmd daemon running on each compute node and a central slurmctld daemon running on a management node (with the possibility of setting up a fail-over twin). If `accounting' is enabled, then there is a third daemon called slurmdbd managing the communications between an accounting database and the management node.

Common slurm user commands include *sacct, salloc, sattach, sbatch, sbcast, scancel, scontrol, sinfo, smap, squeue, srun, strigger and sview* and they are available on each compute node. Manual pages for each of these commands are accessible in the usual manner, that is `man sbatch` for example (see below).

Node

A node in slurm is a compute resource. This is usually defined by particular consumable resources, i.e. memory, CPU, etc…

Partitions

A partition (or queue) is a set of nodes with usually common characteristics and/or limits.

Partitions group nodes into logical (even overlapping if necessary) sets.

Jobs

Jobs are allocations of consumable resources assigned to a user under specified conditions.

Job Steps

A job step is a single task within a job. Each job can have multiple tasks (steps) even parallel ones.

Common user commands

- **sacct**: used to report job or job step accounting information about active or completed jobs.
- **salloc**: used to allocate resources for a job in real time. Typically this is used to allocate resources and spawn a shell. The shell is then used to execute srun commands to launch parallel tasks.
- **sbatch**: used to submit a job script for later execution. The script typically contains an srun command to launch parallel tasks plus any other environment definitions needed.
- **scancel**: used to cancel a pending or running job or job step.
- **sinfo**: used to report the state of partitions and nodes managed by Slurm.
- **squeue**: used to report the state of running and pending jobs or job steps.
- **srun**: used to submit a job for execution or initiate job steps in real time. srun allows users to requests arbitrary consumable resources.

Examples

Determine what partitions exist on the system, what nodes they include, and the general system state.

```
$ sinfo
PARTITION   AVAIL  TIMELIMIT  NODES  STATE NODELIST
playground*    up   infinite      4  alloc maris[029-032]
playground*    up   infinite     32   idle maris[004-022,033,035-046]
lowmem         up 7-00:00:00      1    mix maris047
lowmem         up 7-00:00:00     20   idle maris[048-050,052-068]
lowmem-inf     up   infinite      1    mix maris047
lowmem-inf     up   infinite     20   idle maris[048-050,052-068]
highmem        up 7-00:00:00      6   idle maris[069-074]
highmem-inf    up   infinite      6   idle maris[069-074]
notebook       up   infinite      2    mix maris[023-024]
notebook       up   infinite      4   idle maris[025-028]
```

A * near a partition name indicates the default partition. See `man sinfo`

What jobs exist on the system?

```
$squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
             12277    lowmem CFnumder marxxxel  R    1:23:18      1 maris047
             12276 playgroun pkequal_ maxxxxel  R    1:24:57      1 maris032
              8439  notebook obrien-j   oxxxen  R   18:10:55      1 maris024
              8749 playgroun slurm_co oxxxxxkh  R   17:28:55      1 maris029
              5801  notebook ostroukh oxxxxxkh  R 4-05:02:54      1 maris023
              8750 playgroun slurm_en oxxxxxkh  R   17:28:52      4
maris[029-032]
```

See `man squeue`.

Report more detailed information about partitions, nodes, jobs, job steps, and configuration

```
$ scontrol show partition notebook
PartitionName=notebook
   AllowGroups=ALL AllowAccounts=ALL AllowQos=ALL
   AllocNodes=ALL Default=NO QoS=notebook
   DefaultTime=NONE DisableRootJobs=NO ExclusiveUser=NO GraceTime=0
Hidden=NO
   MaxNodes=UNLIMITED MaxTime=UNLIMITED MinNodes=1 LLN=NO
MaxCPUsPerNode=UNLIMITED
   Nodes=maris0[23-28]
   PriorityJobFactor=1 PriorityTier=1 RootOnly=NO ReqResv=NO
OverSubscribe=NO
   OverTimeLimit=NONE PreemptMode=OFF
   State=UP TotalCPUs=48 TotalNodes=6 SelectTypeParameters=NONE
   DefMemPerNode=UNLIMITED MaxMemPerCPU=4096
```

```
$scontrol show node maris004
NodeName=maris004 Arch=x86_64 CoresPerSocket=4
   CPUAlloc=8 CPUErr=0 CPUTot=8 CPULoad=0.01
   AvailableFeatures=(null)
   ActiveFeatures=(null)
   Gres=(null)
   NodeAddr=maris004 NodeHostName=maris004 Version=16.05
   OS=Linux RealMemory=16046 AllocMem=16000 FreeMem=2082 Sockets=2 Boards=1
   State=ALLOCATED ThreadsPerCore=1 TmpDisk=9951 Weight=1 Owner=N/A
MCS_label=N/A
   BootTime=2016-12-22T12:08:05 SlurmdStartTime=2017-02-17T09:19:46
   CapWatts=n/a
   CurrentWatts=0 LowestJoules=0 ConsumedJoules=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

```
novamaris [1087] $ scontrol show jobs 1052
JobId=1052 JobName=slurm_engine.sbatch
   UserId=xxxxxxx(1261909) GroupId=lorentz(9999) MCS_label=N/A
   Priority=1 Nice=0 Account=zzzzzz QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:49:06 TimeLimit=UNLIMITED TimeMin=N/A
   SubmitTime=2017-02-23T12:17:34 EligibleTime=2017-02-23T12:17:34
   StartTime=2017-02-23T12:17:36 EndTime=Unknown Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=average-computation AllocNode:Sid=maris004:20658
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=maris[024-033,035-040]
   BatchHost=maris024
   NumNodes=16 NumCPUs=128 NumTasks=128 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=128,mem=514784M,node=16
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=32174M MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=./slurm_engine.sbatch
   WorkDir=/marisdata/%u/
   StdErr=/marisdata/xxxxxxx/.log/abcd.err
   StdIn=/dev/null
   StdOut=/marisdata/xxxxxxx/.log/abcd.out
   Power=
```

See `man scontrol`.

Create three tasks running on different nodes

```
novamaris [1088] $ srun -N3 -l /bin/hostname
2: maris007
0: maris005
1: maris006
```

Create three tasks running on the same node

```
novamaris [1090] $ srun -n3 -l /bin/hostname
2: maris005
1: maris005
0: maris005
```

Create three tasks running on different nodes specifying which nodes should at least be used

```
 srun -N3 -w "maris00[5-6]" -l /bin/hostname
1: maris006
0: maris005
2: maris007
```

Allocate resources and spawn job steps within that allocation

```
novamaris [1094] $ salloc -n2
salloc: Granted job allocation 1061
novamaris [997] $ srun /bin/hostname
maris005
maris005
novamaris [998] $ exit
exit
salloc: Relinquishing job allocation 1061
novamaris [1095] $
```

Create a job script and submit it to slurm for execution

 Use `swizard' to generate a batch script.

Or wrote your own script to submit using `sbatch script.sh`.

Less-common user commands

- **sacctmgr**
- **sstat**
- **sshare**
- **sprio**

sacctmgr

Display info on configured qos

```
$ sacctmgr show qos format=Name,MaxCpusPerUser,MaxJobsPerUser,Flags
      Name MaxCPUsPU MaxJobsPU                Flags
---------- --------- --------- --------------------
    normal
playground        32                   DenyOnLimit
  notebook         4         1          DenyOnLimit
```

sstat

Displays information pertaining to CPU, Task, Node, Resident Set Size (RSS) and Virtual Memory (VM) of a running job

```
$sstat -o JobID,MaxRSS,AveRSS,MaxPages,AvePages,AveCPU,MaxDiskRead
8749.batch

       JobID     MaxRSS     AveRSS MaxPages   AvePages     AveCPU
```

```
MaxDiskRead
------------ ---------- ---------- -------- ---------- ---------- ----------
--
8749.batch       196448K    196448K        0          0  01:00.000 7.03M
```

 If your job is serial (not parallel, that is not submitted using `srun') do not forget to append `.batch` to the job id.

 For parallel jobs `sstat <jobid>` will work.

sshare

Display the shares associated to a particular user

```
$ sshare -U -u xxxxx
             Account       User  RawShares  NormShares    RawUsage
EffectvUsage  FairShare
-------------------- ---------- ---------- ----------- ----------- ---------
---- ----------
xxxxx                   yyyyyy         1    0.055556      691078
0.005142   0.937857
```

 On maris, usage parameters will decay over time according to a PriorityDecayHalfLife of 14 days.

sprio

Display priority information of a pending job id xxx

```
sprio -l -j xxx
```

Tips

To minimize the time your job spends in the queue you could specify multiple partitions so that the job can start as soon as possible. Use `–partition=notebook,playground,lowmem`

To have a rough estimate of when your queued job will start type `squeue –start`

To translate a job script written for a scheduler different than slurm to slurm's own syntax consider using http://www.schedmd.com/slurmdocs/rosetta.pdf

top-like node usage

Should you want to monitor the usage of the cluster nodes in a top-like fashion type

```
watch -n 1 -x sinfo -S"-O" -o "%.9n %.6t %.10e/%m %.10O %.15C"
```

top-like job stats

To monitor the resources consumed by your running job type

```
watch -n1 sstat --format
JobID,NTasks,nodelist,MaxRSS,MaxVMSize,AveRSS,AveVMSize,AveCpuFreq
<jobid>[.batch]
```

Make local file available to all nodes allocated to a slurm job

To transmit a file to all nodes allocated to the currently active Slurm job use `sbcast`. For instance

```
 > cat my.job
   #!/bin/env bash
   sbcast my.prog /tmp/my.prog
   srun /tmp/my.prog

 > sbatch --nodes=8 my.job
   srun: jobid 145 submitted
```

Specify nodes for a job

For instance `#SBATCH –nodelist=maris0xx`

Environment variables available to any slurm job

You can use any of the following variables in your jobs

```
$ salloc -p playground -N 10
salloc: Granted job allocation 13709
$  printenv | grep -i slurm_
SLURM_NODELIST=maris[031-033,035-041]
SLURM_JOB_NAME=bash
SLURM_NODE_ALIASES=(null)
SLURM_JOB_QOS=normal
SLURM_NNODES=10
SLURM_JOBID=13709
SLURM_TASKS_PER_NODE=1(x10)
```

```
SLURM_JOB_ID=13709
SLURM_SUBMIT_DIR=/tmp/bla-bla
SLURM_JOB_NODELIST=maris[031-033,035-041]
SLURM_CLUSTER_NAME=maris
SLURM_JOB_CPUS_PER_NODE=1(x10)
SLURM_SUBMIT_HOST=novamaris.lorentz.leidenuniv.nl
SLURM_JOB_PARTITION=playground
SLURM_JOB_ACCOUNT=yuyuysu
SLURM_JOB_NUM_NODES=10
SLURM_MEM_PER_NODE=32174
```