Running Jobs on Maxwell

---

schedmd offers exhaustive documentation how to use SLURM: http://slurm.schedmd.com/. We have just collected a few examples below.

Maxwell useful commands provides a short list of commands which might become handy.

General remarks

Currently all nodes are configured as non-shared resources. A job will have exclusive access to the resources requested, and it's up to the job to consume all CPU-cores or a subset of it, using mpi, p-threads or forking processes and so on. It consequently doesn't make sense to use the --ntasks or --cpus_per_task setting. Also try to avoid using specific hostnames; there is no advantage and tends to be counter-productive. All you need to specify are the partition, the runtime of your job and the number of nodes:

```
#SBATCH --partition=maxwell    # this is the default partition.
#SBATCH --time=00:10:00        # default is 1h. The maximum is partition dependent, have a look at sview or
scontrol for details.
#SBATCH --nodes=1              # Number of nodes. If your job can consume variable number of nodes you
might want to use something like
#SBATCH --nodes=2-6            # which requests a minimum of 2 and a maximum of 6 nodes.
```

See the FAQ and the schedmd documentation for more details. To get a quick overview on a commands syntax use the man pages (available on any of the slurm login nodes): man <command>. You will most likely need not more than a handful of commands, like salloc, sbatch, scancel, scontrol, sinfo and possibly sview for a gui,

Running Jobs in Batch

simple batch job using wrap option to launch command

```
# simple job which prints hostname
[@max-wgs ~]$ sbatch --wrap hostname
Submitted batch job 1516

[@max-wgs ~]$ ls
slum-1516.out

[@max-wgs ~]$ cat slurm-1516.out
max-wgs.desy.de
```

batch script submission using command line options

```
# simple job which prints hostname
[@max-wgs ~]$ cat hostname.sh
#!/bin/bash
#SBATCH --partition=maxwell
#SBATCH --time=00:10:00                          # Maximum time requested
#SBATCH --nodes=1                                # Number of nodes
#SBATCH --chdir   /home/mmuster/slurm/output     # directory must already exist!
#SBATCH --job-name  hostname
#SBATCH --output    hostname-%N-%j.out           # File to which STDOUT will be written
#SBATCH --error     hostname-%N-%j.err           # File to which STDERR will be written
#SBATCH --mail-type END                          # Type of email notification- BEGIN,END,FAIL,ALL
#SBATCH --mail-user max.muster@desy.de           # Email to which notifications will be sent. It defaults
to <userid@mail.desy.de> if none is set.

/bin/hostname

# submit to batch queue for one node with one task
# requesting 10 mins of wall time
[@max-wgs ~]$ sbatch hostname.sh
Submitted batch job 2163

[@max-wgs ~]$ ls
hostname.sh  slurm-2163.out

[@max-wgs ~]$ cat slurm-2163.out
max-wn004.desy.de

[@max-wgs ~]$ scontrol show job 2163
JobId=2163 JobName=hostname
   UserId=mmuster(1234) GroupId=cfel(3512)
   Priority=5001 Nice=0 Account=cfel QOS=cfel
   JobState=COMPLETED Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:01 TimeLimit=00:10:00 TimeMin=N/A
   SubmitTime=2016-01-20T14:50:17 EligibleTime=2016-01-20T14:50:17
   StartTime=2016-01-20T14:50:17 EndTime=2016-01-20T14:50:18
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=cfel AllocNode:Sid=max-cfel001:1345
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=max-cfel004
   BatchHost=max-cfel004
   NumNodes=1 NumCPUs=64 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=64,node=1
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   Shared=0 Contiguous=0 Licenses=(null) Network=(null)
   Command=/home/mmuster/slurm/hostname.sh
   WorkDir=/home/mmuster/slurm/output
   StdErr=/home/mmuster/slurm/output/hostname-%N-2163.err
   StdIn=/dev/null
   StdOut=/home/mmuster/slurm/output/hostname-%N-2163.out
   Power= SICP=0
```

Testing Jobs

You can test in advance if the resources requested by your job are available at all, and when the job is expected to start. For example

```
# request a single node without particular specs
[@max-wgs001 ~]$ sbatch --test-only my-app.sh  # --test-only indicates a dry-run. It won't submit the job
sbatch: Job 1652551 to start at 2019-01-20T09:29:06 using 32 processors on nodes max-wn004 in partition
maxwell

# request a V100 GPU
[@max-wgs001 ~]$ sbatch  --constraint=V100 --test-only my-app.sh
sbatch: Job 1652553 to start at 2019-01-20T09:29:26 using 40 processors on nodes max-wng019 in partition
maxwell

# request 2 V100 GPUs in a single node, which is an invalid constraint
[@max-wgs001 ~]$ sbatch --constraint="GPUx2&V100" --test-only my-app.sh
allocation failure: Requested node configuration is not available

# ask for a P100 OR V100 in one of the partitions:
[@max-wgs001 ~]$ sbatch --partition="all,petra4,upex" --constraint="V100|P100" --test-only my-app.sh
sbatch: Job 1652560 to start at 2019-01-18T11:34:15 using 40 processors on nodes max-exflg006 in partition
upex
# having neither permission to use petra4 or upex my job must be running in the all partition.
# --test-only does not validate the partition proposed. Make sure to specify only partitions usable for
your account.

# to verify:
[@max-wgs001 ~]$ sbatch --partition="all,petra4,upex" --constraint="V100|P100"  my-app.sh
Submitted batch job 1652561
[schluenz@max-wgs001 ~]$ sacct -j 1652561
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
1652561       my-app.sh        all         it         40  COMPLETED      0:0
# as expected running in all partition.
```

Running Jobs Interactively

You can make an interactive reservation using salloc:

```
[@max-wgs ~]$ salloc -N 4 --partition=maxwell                # allocate 4 nodes on partition maxwell. See
"Groups & Partitions" for more information
salloc: Granted job allocation 214

[@max-wgs ~]$ scontrol show -d job 214                       # show what you've got
   NodeList=max-wn[003-006]
   BatchHost=max-wn003
   NumNodes=4 NumCPUs=256 CPUs/Task=1 ReqB:S:C:T=0:0:*:*

# while your allocation is active you can run for example an MPI-job interactively.
# But you can also login to one of the allocated nodes:
[@max-wgs ~]$ max-wn003
[@max-wn003 ~]$ mpirun ...
[@max-wn003 ~]$ exit     # terminate the ssh session. It does NOT release the allocation!

# remember to release the allocation once done!
[@max-wgs ~]$ exit
exit
salloc: Relinquishing job allocation 214
salloc: Job allocation 214 has been revoked.
```

salloc spawns a shell and sets the SLURM specific environment. As long as the shell is active (and the time-limit not exceeded), the resources are allocated. Leaving the shell returns the resources to the pool.

Running graphical applications interactively

```
[@max-wgs ~]$ salloc -N 1 --partition=maxwell
salloc: Granted job allocation 214

[@max-wgs ~]$ ssh -t -Y $SLURM_JOB_NODELIST matlab_R2018a                  # this will work on max-wgs,
but crash on max-display!
[@max-wgs ~]$ ssh -t -Y $SLURM_JOB_NODELIST matlab_R2018a -softwareopengl   # this will always work

# you could write a small wrapper named $HOME/bin/s:
#!/bin/bash
if [[ "x$SLURM_JOB_NODELIST" != "x" ]]; then
    ssh -t -Y $SLURM_JOB_NODELIST "$@"
else
    echo "salloc -N 1 before using s!"
fi

[@max-wgs ~]$ s matlab_R2018a -softwareopengl

# remember to release the allocation once done!
[@max-wgs ~]$ exit
exit
salloc: Relinquishing job allocation 214
salloc: Job allocation 214 has been revoked.
```

Controlling Jobs

```
scancel <jobid>                  # cancel a job
scancel -u <username>            # cancel all the jobs for a user
scancel -t PENDING -u <username>  # cancel all the pending jobs for a user
scancel --name myJobName         # cancel one or more jobs by name
scontrol hold <jobid>            # pause a particular job
scontrol resume <jobid>          # resume a particular job
scontrol requeue <jobid>         # requeue (cancel and rerun) a particular job
```

Job Information

To get a quick overview about jobs, partitions and so on, slurm provides a tool sview:

how to cancel a job

```
[@max-wgs ~]$ sview &
```

By default it will only show partitions you are entitled to submit jobs to, and consequently only jobs running in these partitions. To view all jobs on all partitions enable under Options: Show Hidden.

```
squeue -u <username>                                                    # List all current jobs for a
user
squeue -u <username> -t RUNNING                                         # List all running jobs for a
user
squeue -u <username> -t PENDING                                         # List all pending jobs for a
user
squeue -u <username> -p maxwell                                         # List all current jobs in the
maxwell partition for a user
scontrol show jobid -dd <jobid>                                         # List detailed information
for a job (useful for troubleshooting)
scontrol update jobid=<jobid> partition=maxwell NumNodes=6              # update a (pending) job for
example by setting the number of nodes compliant with partition-limits

# Once your job has completed, you can get additional information that was not available during the run.
This includes run time, memory used, etc.
sacct -j <jobid> --format=JobID,JobName,MaxRSS,Elapsed                  # To get statistics on
completed jobs by jobID:
sacct -u <username> --format=JobID,JobName,MaxRSS,Elapsed               # To view the same information
for all jobs of a user
```

```
[@max-wgs ~]$ sstat --format=AveCPU,AvePages,AveRSS,AveVMSize,JobID -j 1652579   # some stats about running
job like memory consumption.
    AveCPU   AvePages     AveRSS  AveVMSize        JobID
---------- ---------- ---------- ---------- ------------
 00:00.000          0      1469K     27840K 1652579.0


# /software/tools/bin/slurm is a convenient tool to extract queue and job information
[@max-wgs ~]$ module load maxwell tools
[@max-wgs ~]$ slurm
Show or watch job queue:
 slurm [watch] queue     show own jobs
 slurm [watch] q <user>  show user's jobs
 slurm [watch] quick     show quick overview of own jobs
 slurm [watch] shorter   sort and compact entire queue by job size
 slurm [watch] short     sort and compact entire queue by priority
 slurm [watch] full      show everything
 slurm [w] [q|qq|ss|s|f] shorthands for above!

 slurm qos               show job service classes
 slurm top [queue|all]   show summary of active users

Show detailed information about jobs:
 slurm prio [all|short]  show priority components
 slurm j|job <jobid>     show everything else
 slurm steps <jobid>     show memory usage of running srun job steps

Show usage and fair-share values from accounting database:
 slurm h|history <time>  show jobs finished since, e.g. "1day" (default)
 slurm shares

Show nodes and resources in the cluster:
 slurm p|partitions      all partitions
 slurm n|nodes           all cluster nodes
 slurm c|cpus            total cpu cores in use
 slurm cpus <partition>  cores available to partition, allocated and free
 slurm cpus jobs         cores/memory reserved by running jobs
 slurm cpus queue        cores/memory required by pending jobs
 slurm features          List features and GRES
 slurm brief_features    List features with node counts
 slurm matrix_features   List possible combinations of features with node counts

# example: show jobs
slurm q
JOBID              PARTITION NAME                 TIME       START_TIME    STATE NODELIST(REASON)
1599696            maxwell   test1                0:00              N/A  PENDING (QOSMaxJobsPerUserLimit)
1599695            maxwell   test2                0:00              N/A  PENDING (QOSMaxJobsPerUserLimit)
1599689            maxwell   test3             3:53:05 2019-01-18T08:02  RUNNING max-wn[017,026],max-wna
[022-025]
# slurm w q would continously update the view on your jobs
```

Resources

Requesting specific resources can largely be done using constraints. A full list of available constraints can be found on the hardware page and combination of constraints page. Supported constraints are

```
#SBATCH --constraint=AMD             # request AMD-nodes, the former it-hpc-nodes.
#SBATCH --constraint=GPU             # request nodes with GPUs.
#SBATCH --constraint="GPUx1&V100"    # request a node with exactly one NVIDIA V100 GPU.
#SBATCH --constraint=INTEL           # request intel nodes.
#SBATCH --constraint="INTEL&V3"      # request intel nodes with v3 CPUs (haswell). V2 (IvyBridge) are still
in use on the GPU nodes. v4 (Broadwell) will come soon.
#SBATCH --constraint="[AMD|INTEL]"   # request either N*INTEL or N*AMD nodes. nodes will be uniform.
Without [] any combination of INTEL and AMD nodes is being requested.
```

Environment

```
[@max-wgs001 ~]$ salloc -N 1 -J test         # request a single node in the default
partition

salloc: Granted job allocation
3327

salloc: Waiting for resource
configuration

salloc: Nodes max-wn007 are ready for job    # the host(s) allocated. You can ssh into the node, even from
a different host (e.g. your windows pc)

[@max-wgs001 ~]$ env | grep SLURM            # show environment
SLURM_SUBMIT_DIR=/home/schluen
SLURM_SUBMIT_HOST=max-wgs001.desy.de
SLURM_JOB_ID=3327
SLURM_JOB_NAME=test
SLURM_JOB_NUM_NODES=1
SLURM_JOB_NODELIST=max-wn007
SLURM_NODE_ALIASES=(null)
SLURM_JOB_PARTITION=maxwell
SLURM_JOB_CPUS_PER_NODE=32
SLURM_JOBID=3327
SLURM_NNODES=1
SLURM_NODELIST=max-wn007
SLURM_TASKS_PER_NODE=32
SLURM_CLUSTER_NAME=maxwell
[max-wgs001 ~]$ exit                        # return resources
exit
salloc: Relinquishing job allocation 3327
salloc: Job allocation 3327 has been revoked.
```

The list of nodes is actually represented as a range. If you need a regular hostlist a scriplet like (see https://rc.fas.harvard.edu/resources/running-jobs/)

```
#!/bin/bash
hostlist=$(scontrol show hostname $SLURM_JOB_NODELIST)
rm -f hosts

for f in $hostlist
  do
  echo $f':64' >> hosts
done
```

should do. Now you can use the 256 cores in an mpi job:

```
[@max-wgs ~]$ mpirun -n 256 hello-mpi    # would give you 256 lines of "Hello world" back
Hello world from processor max-wn005.desy.de, rank 165 out of 256 processors
Hello world from processor max-wn004.desy.de, rank 124 out of 256 processors
# Note: since mpirun knows about the slurm allocation it will use the allocated hosts, not the local host!
```

OpenMPI would know about the hosts to use. However, running an application like mathematica interactively would still use the WGS you initially used to make the allocation with salloc; but while your allocation is valid, you can connect to any host allocated with ssh and run for example mathematica kernels across the nodes allocated.

Making advance reservations

Advance reservation is currently not possible for users.

Jobs with Dependencies

Jobs can be chained in a way that one jobs doesn't start before a set of jobs hasn't reached a particular state. Most commonly is probably to start a jobs only after some other job has finished:

```
[@max-wgs ~]$ sbatch --dependency=afterok:1234:1235  dep1.sh # start dep1.sh only after jobs with jobid
1234 and 1235 have finished successfully.


[@max-wgs ~]$ sbatch --dependency=singleton --job-name=singleton singleton.sh    #  start this job only
after all jobs with the same job-name have finished.
                                                                                 #  makes sure that only a
single job of this name can run at a time (for a particular user)
```

Array Jobs

Array jobs allow to launch a set of identical, indexed jobs. Lets assume you want to process 10 images with identical environment:

```
# array-job.sh
#!/bin/bash
#SBATCH --time    0-00:01:00
#SBATCH --nodes            1
#SBATCH --partition all
#SBATCH --array 1-10
#SBATCH --job-name job-array
#SBATCH --output array-%A_%a.out
export LD_PRELOAD=""
source /etc/profile.d/modules.sh
echo "SLURM_JOB_ID           $SLURM_JOB_ID"
echo "SLURM_ARRAY_JOB_ID     $SLURM_ARRAY_JOB_ID"
echo "SLURM_ARRAY_TASK_ID    $SLURM_ARRAY_TASK_ID"
echo "SLURM_ARRAY_TASK_COUNT $SLURM_ARRAY_TASK_COUNT"
echo "SLURM_ARRAY_TASK_MAX   $SLURM_ARRAY_TASK_MAX"
echo "SLURM_ARRAY_TASK_MIN   $SLURM_ARRAY_TASK_MIN"

process image_${SLURM_ARRAY_TASK_ID}.tif


[@max-wgs ~]$ sbatch array-job.sh
```