Modules and Slurm

Access to Softwares : Modules

Preinstalled softwares can be listed, enabled or disabled through the use of the module command

- `module avail`: list available software
- `module load`: set up the environment to use the software
- `module list`: list currently loaded software
- `module purge`: clears the environment

Modules

The main module of interest for us is the GCC compiler

```
 $ module av gcc/ GCC/
------------------- Releases ( 2019b ) --------------------
   GCC/8.3.0
 $ module load GCC/8.3.0
 $ module list
Currently Loaded Modules:
...  3) StdEnv        (H) 5) zlib/1.2.11-GCCcore-8.3.0   7) GCC/8.3.0
...  4) GCCcore/8.3.0     6) binutils/2.32-GCCcore-8.3.0
```

Modules

We look for the `GCC` module using the `module av` command. The result of the command tell us that there is a module named `GCC/8.3.0` available.

```
 $ module av gcc/ GCC/
------------------- Releases ( 2019b ) -------------------
   GCC/8.3.0
 $ module load GCC/8.3.0
 $ module list
Currently Loaded Modules:
...  3) StdEnv        (H) 5) zlib/1.2.11-GCCcore-8.3.0   7) GCC/8.3.0
...  4) GCCcore/8.3.0     6) binutils/2.32-GCCcore-8.3.0
```

Modules

We load the `GCC/8.3.0` module with the `module load` command.

```
 $ module av gcc/ GCC/
------------------- Releases ( 2019b ) --------------------
   GCC/8.3.0
 $ module load GCC/8.3.0
 $ module list
Currently Loaded Modules:
...  3) StdEnv        (H) 5) zlib/1.2.11-GCCcore-8.3.0   7) GCC/8.3.0
...  4) GCCcore/8.3.0     6) binutils/2.32-GCCcore-8.3.0
```

Modules

Using the `module list` command we can see the `GCC` module is now loaded as well as its dependencies

```
 $ module av gcc/ GCC/
-------------------- Releases ( 2019b ) --------------------
   GCC/8.3.0
 $ module load GCC/8.3.0
 $ module list
Currently Loaded Modules:
...  3) StdEnv        (H) 5) zlib/1.2.11-GCCcore-8.3.0   7) GCC/8.3.0
...  4) GCCcore/8.3.0     6) binutils/2.32-GCCcore-8.3.0
```

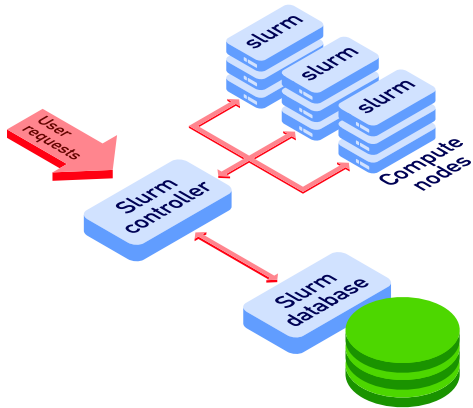
Basics of Slurm

Resource Sharing on a Supercomputer

- Resource sharing on a supercomputer dedicated to scientific computing is often organized by a piece of software called a resource manager or job scheduler.
- Users submit jobs, which are scheduled and allocated resources (CPU time, memory, ...) by the resource manager.

Resource Sharing on a Supercomputer

The CÉCI clusters job scheduler is Slurm.



The main Slurm Commands

- `sinfo`: view information about Slurm nodes and partitions
- `squeue`: view information about jobs located in the Slurm scheduling queue
- `sbatch`: submit a batch script to Slurm
- `scancel`: Cancel a job

sinfo

View information about Slurm nodes and partitions.

Here we have two partitions, `batch` and `debug` with maximum allowed run time of 2 days and 6 hours respectively. These two partitions are available for computing (`up`).

```
 $ sinfo
CLUSTER: lemaitre3
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
batch*       up 2-00:00:00      8    mix lm3-w[003,005,038, ...]
batch*       up 2-00:00:00     70  alloc lm3-w[004,006-037, ...]
debug        up    6:00:00      4  alloc lm3-w[091-094]
```

sinfo

The first line corresponds to the nodes in the `batch` partition that are in a `mix` state. This means that all the resources available on the node listed in the last column are not fully allocated.

```
 $ sinfo
CLUSTER: lemaitre3
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
batch*       up 2-00:00:00      8    mix lm3-w[003,005,038, ...]
batch*       up 2-00:00:00     70  alloc lm3-w[004,006-037, ...]
debug        up    6:00:00      4  alloc lm3-w[091-094]
```

sinfo

The second line corresponds to the nodes in the `batch` partition that are in an `alloc` state. This means that all the resources available on the node listed in the last column are allocated.

```
 $ sinfo
CLUSTER: lemaitre3
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
batch*       up 2-00:00:00      8    mix lm3-w[003,005,038, ...]
batch*       up 2-00:00:00     70  alloc lm3-w[004,006-037, ...]
debug        up    6:00:00      4  alloc lm3-w[091-094]
```

sinfo

The third line corresponds to the nodes in the `debug` partition that are in an `alloc` state. This means that all the resources available on the node listed in the last column are allocated.

```
 $ sinfo
CLUSTER: lemaitre3
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
batch*       up 2-00:00:00      8    mix lm3-w[003,005,038, ...]
batch*       up 2-00:00:00     70  alloc lm3-w[004,006-037, ...]
debug        up    6:00:00      4  alloc lm3-w[091-094]
```

squeue

View information about jobs located in the Slurm scheduling queue

```
 $ squeue
CLUSTER: lemaitre3
   JOBID PARTITION     NAME     USER ST    TIME  NODES NODELIST(REASON)
69093677     batch bouncing  bahardy PD    0:00     43 (Resources)
69084075     batch PYV3_wB_ fmairess PD    0:00     12 (Priority)
...
69093589     batch x1000_6_     jlam  R   23:21      2 lm3-w[053,065]
69093066     batch   corona  nfranco  R   13:09      1 lm3-w080
```

squeue

This job is in a pending state (PD). This means that the job is waiting for the scheduler to grant permission to start. The reason why this job is waiting is the lack of `resources`. This is not surprising as the user requested 43 computes nodes.

```
 $ squeue
CLUSTER: lemaitre3
   JOBID PARTITION     NAME     USER ST    TIME  NODES NODELIST(REASON)
69093677     batch bouncing  bahardy PD    0:00     43 (Resources)
69084075     batch PYV3_wB_ fmairess PD    0:00     12 (Priority)
...
69093589     batch x1000_6_     jlam  R   23:21      2 lm3-w[053,065]
69093066     batch   corona  nfranco  R   13:09      1 lm3-w080
```

squeue

This job is in a pending state (PD) too. The reason why this job is waiting is that the `priority` of the user is to low. The scheduler run job with higher priority first.

```
 $ squeue
CLUSTER: lemaitre3
   JOBID PARTITION     NAME     USER ST    TIME  NODES NODELIST(REASON)
69093677     batch bouncing  bahardy PD    0:00     43 (Resources)
69084075     batch PYV3_wB_ fmairess PD    0:00     12 (Priority)
...
69093589     batch x1000_6_     jlam  R   23:21      2 lm3-w[053,065]
69093066     batch   corona  nfranco  R   13:09      1 lm3-w080
```

squeue

This job is running (R) for 23 hours and 21 minutes. It has been allocated 2 compute nodes (lm3-w053 and lm3-w065)

```
 $ squeue
CLUSTER: lemaitre3
   JOBID PARTITION     NAME     USER ST    TIME  NODES NODELIST(REASON)
69093677     batch bouncing  bahardy PD    0:00     43 (Resources)
69084075     batch PYV3_wB_ fmairess PD    0:00     12 (Priority)
...
69093589     batch x1000_6_     jlam  R   23:21      2 lm3-w[053,065]
69093066     batch   corona  nfranco  R   13:09      1 lm3-w080
```

squeue

This job is running (R) for 13 hours and 9 minutes. It has been allocated 1 compute nodes (lm3-w080)

```
 $ squeue
CLUSTER: lemaitre3
   JOBID PARTITION     NAME     USER ST    TIME  NODES NODELIST(REASON)
69093677     batch bouncing  bahardy PD    0:00     43 (Resources)
69084075     batch PYV3_wB_ fmairess PD    0:00     12 (Priority)
...
69093589     batch x1000_6_     jlam  R   23:21      2 lm3-w[053,065]
69093066     batch   corona  nfranco  R   13:09      1 lm3-w080
```

squeue

You can use `squeue` with the `-u` option and your username to only see your jobs

```
  $ squeue -u user
CLUSTER: lemaitre3
   JOBID PARTITION     NAME     USER ST     TIME  NODES NODELIST(REASON)
69071111     batch     job1     user PD     0:00      6 (Priority)
69084075     batch     job2     user PD     0:00     12 (Priority)
69071129     batch     job3     user PD     0:00      4 (Priority)
69084074     batch     job4     user PD     0:00     10 (Priority)
```

sbatch

The `sbatch` command allows you to submit a batch script to Slurm. This script is a file with specific commands to Slurm as well as commands to execute on the compute nodes.

```
 $ sbatch your_job_script.sh
Submitted batch job 69095263 on cluster lemaitre3
```

sbatch

In a submission script, lines prefixed with `#SBATCH` and followed by a command are understood by Slurm as resource requests.

`--time`	Limit on the run time of the job
`--ntasks`	Maximum of number tasks (MPI ranks)
`--cpus-per-task`	Number of processors per task (threads)
`--mem-per-cpu`	Minimum memory required per allocated CPU
`--partition`	Request a specific partition for the resource allocation

Typical Script for an OpenMP Program

```
#!/bin/bash
# Submission script for Lemaitre3
#SBATCH --time=0-01:00:00 # dd-hh:mm:ss
#
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1024 # megabytes
#SBATCH --partition=batch

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

module load GCCcore/8.3.0
cd $SLURM_SUBMIT_DIR

./your_program
```

Typical Script for an OpenMP Program

```
#!/bin/bash
# Submission script for Lemaitre3
#SBATCH --time=0-01:00:00 # dd-hh:mm:ss
#
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1024 # megabytes
#SBATCH --partition=batch

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

module load GCCcore/8.3.0
cd $SLURM_SUBMIT_DIR

./your_program
```

You have to start your script with this line

Typical Script for an OpenMP Program

```
#!/bin/bash
# Submission script for Lemaitre3
#SBATCH --time=0-01:00:00 # dd-hh:mm:ss
#
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1024 # megabytes
#SBATCH --partition=batch

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

module load GCCcore/8.3.0
cd $SLURM_SUBMIT_DIR

./your_program
```

Maximum run time: 1 hour

Typical Script for an OpenMP Program

```
#!/bin/bash
# Submission script for Lemaitre3
#SBATCH --time=0-01:00:00 # dd-hh:mm:ss
#
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1024 # megabytes
#SBATCH --partition=batch

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

module load GCCcore/8.3.0
cd $SLURM_SUBMIT_DIR

./your_program
```

Only one task (one process)

Typical Script for an OpenMP Program

```
#!/bin/bash
# Submission script for Lemaitre3
#SBATCH --time=0-01:00:00 # dd-hh:mm:ss
#
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1024 # megabytes
#SBATCH --partition=batch

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

module load GCCcore/8.3.0
cd $SLURM_SUBMIT_DIR

./your_program
```

Typical Script for an OpenMP Program

```
#!/bin/bash
# Submission script for Lemaitre3
#SBATCH --time=0-01:00:00 # dd-hh:mm:ss
#
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1024 # megabytes
#SBATCH --partition=batch

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

module load GCCcore/8.3.0
cd $SLURM_SUBMIT_DIR

./your_program
```


Memory per cpu/core

Typical Script for an OpenMP Program

```
#!/bin/bash
# Submission script for Lemaitre3
#SBATCH --time=0-01:00:00 # dd-hh:mm:ss
#
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1024 # megabytes
#SBATCH --partition=batch

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

module load GCCcore/8.3.0
cd $SLURM_SUBMIT_DIR

./your_program
```

Typical Script for an OpenMP Program

```
#!/bin/bash
# Submission script for Lemaitre3
#SBATCH --time=0-01:00:00 # dd-hh:mm:ss
#
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1024 # megabytes
#SBATCH --partition=batch

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

module load GCCcore/8.3.0
cd $SLURM_SUBMIT_DIR

./your_program
```

Set the number of OpenMP threads

Typical Script for an OpenMP Program

```
#!/bin/bash
# Submission script for Lemaitre3
#SBATCH --time=0-01:00:00 # dd-hh:mm:ss
#
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1024 # megabytes
#SBATCH --partition=batch

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

module load GCCcore/8.3.0
cd $SLURM_SUBMIT_DIR

./your_program
```

← Run your program

That’s All Folks!