Computing at CSUC: A Quick Guide

09-04-2022

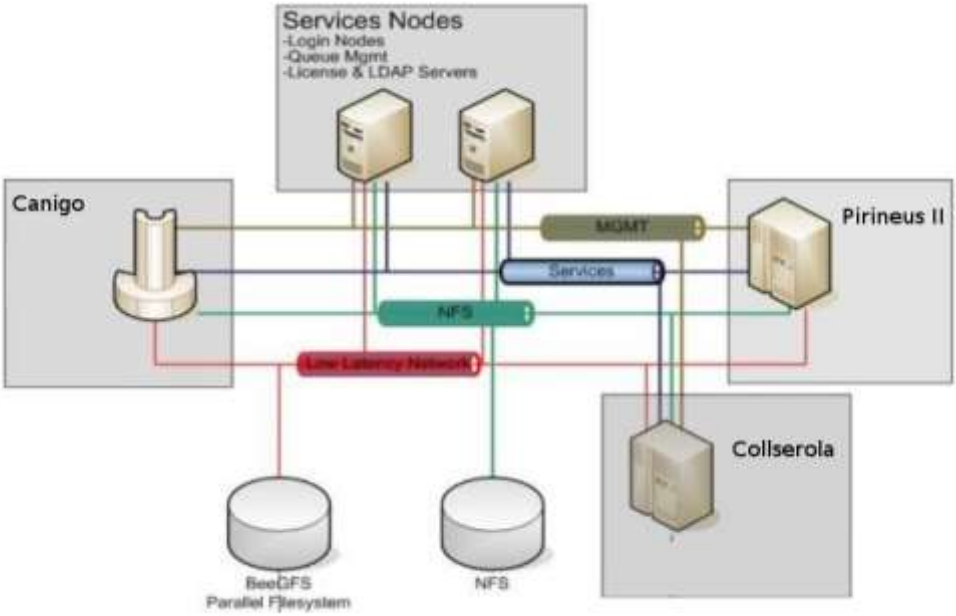
System description

The computing facilities at CSUC are composed of two different systems, but configured as a unique cluster managed by Slurm Workload Manager. With this setup, the batch system oversees which computing node will execute the jobs, depending on the demanded resources, unless the user specifies a node in the job submission. Therefore, jobs can be submitted from the login node, and will be dispatched at the proper computing node as soon as resources will be available.

In the following table we present a short description of the computing and storage resources:

Machine	Description
pirineus II	44 standard nodes with: – 2 Intel Xeon Platinum 8168: 24 cores, 2.7 GHz – 192 GB of main memory – 4 TB local disk 19 standard nodes with: – 2 Intel Xeon Platinum 8268: 24 cores, 2.9 GHz – 192 GB of main memory – 4 TB local disk 6 High memory nodes with: – 2 Intel Xeon Platinum 8168: 24 cores, 2.7 GHz – 384 GB of main memory – 4 TB local disk 4 GPGPU nodes with: – 2 Intel Xeon Platinum 8168: 24 cores, 2.7 GHz – 192 GB of main memory – 4 TB local disk – 2 Nvidia P100 GPGPU (3584 Cuda cores, 12 GB) 4 Intel Knight's Landing nodes with: – 1 Intel Xeon Phi 7250: 68 cores, 1.4 GHz – 384 GB of main memory – 4 TB local disk

Machine	Description
canigo	Currently this system is composed by 2 nodes with: – 8 Intel Xeon Platinum 8168: 24 cores, 2,7 GHz – 3 TB of main memory – 22 TB local disk
biosca	BeeGFS storage cluster, 4 nodes with: – 18 SATA disks with 4 TB each – High speed / low latency InfiniBand EDR network – Raid driver Intel RS3DC080



All the nodes are interconnected and with the BeeGFS cluster by an InfiniBand EDR high speed/low latency connection which provides a 100 Gb/s bandwidth. In addition to these there is also a 10 Gbit Ethernet network providing access to the users' home.

The users' home directories are offered by a NetApp FAS8020 NFS server and the shared scratch by the BeeGFS cluster.

First steps

Log in

Accessing to the system can be achieved by using **ssh** protocol, **through port number 2122**, to the login nodes:

```
ssh -p 2122 <user>@hpc.csuc.cat
```

If it is the first-time connection, the system will require you a password reset. The new password must fulfil the following requisites:

- Minimum of 8 characters.
- Minimum of 1 capital letter [A-Z]
- Minimum of 1 number [0-9]
- Minimum of 1 punctuation sign [. , ; : ? ¿ ! ¡ \ / - _ + * { } ( )].

If you are experiencing trouble logging into the system or in case you do not remember the password, please contact CSUC support staff at **https://hpc.csuc.cat** (registration needed) or at phone number 935 51 62 22.

Transferring files

In order to transfer files to/from CSUC systems you can use ***scp*** command from your local machine (and not to it, as outgoing ssh connections from CSUC systems are not allowed):

Upload:

```
scp -P 2122 <local_file> <user>@hpc.csuc.cat:<remote_directory>
```

Download:

```
scp -P 2122 <user>@hpc.csuc.cat:<remote_file> <local_directory>
```

Alternatively, ***sftp*** can also be used:

```
sftp -oPort=2122 <user>@hpc.csuc.cat
```

Working environment

Directories

The working areas available at CSUC facilities are summarized below:

Name	Variable	Availability	Quota	Time limit	Backup
/home/$USER	$HOME	Global	25 - 200 GB (*)	Unlimited	Yes
/scratch/$USER/	-	Global	1 TB	30 days	No
/scratch/$USER/tmp/$JOBID	$SCRATCH / $SHAREDSCRATCH	Global	1 TB	7 days	No
/tmp/$USER/$JOBID	$SCRATCH / $LOCALSCRATCH	Local node	-	Job execution	No

* Group-wide quota, depending on the basic block acquired.

The environment variable $SCRATCH points to $SHAREDSCRATCH for multi-node jobs and to $LOCALSCRATCH for one-node jobs.

As logging onto a node is only allowed if the user has a job running on it, when a one-node job finishes, the content inside /tmp/$USER/$JOBID is automatically moved to /scratch/$USER/tmp/$JOBID.

Modules

We use Tcl modules to set up the working environment in the HPC infrastructure, in order to see all the available modules that you can use you can execute the following command:

```
module av
```

You will see that there is a structure in the modules:

- All library modules are under ***libs***, for example: FFTW, LAPACK, MKL, etc.
- All compilers and scripting languages (ex. Python) are under ***tools***.
- All the HPC Applications compiled by us are under ***apps***.
- In addition to that we have defined some toolchains that load the minimum set of tools to build a lot of different applications, you can find it under ***toolchains***.

Examples:

1. If you want to use the g16b1 version of Gaussian you can load the corresponding module like:

```
module load apps/gaussian/g16b1
```

2. If you need to compile your own application and you need to use GCC compilers, LAPACK and FFTW libraries you should execute:

```
module load tools/gcc/8.1.0
module load libs/gnu/lapack/3.8.0
module load libs/gnu/fftw/3.3.8
```

3. If you need to compile your own application and you need to use Intel compilers with MKL libraries and OpenMPI libraries, you can do:

```
module load tools/intel/comp_xe_2018
module load mpi/intel/openmpi/3.1.0
module load libs/mkl/2018
```

or

```
module load toolchains/intel_mkl_ompi
```

Jobs

Partitions

In order to improve the scheduling and reduce job's waiting time, a structure of partitions (formerly queues) has been defined. The following table summarizes all the most important partitions features.

Queue	Runtime Limit	Memory (MB/core)	Processors (core/node)	Threads (threads/core)	Observations
std	No limits	Max. 3900	48	1	
std-fat	No limits	3900 - 7900	48	1	
mem	No limits	7900 - 24180	192	1	
knl	No limits	5600	68	4	Whole node allocated
gpu	No limits	Max. 3900	48 + 2 GPGPU	1	24 cores per GPGPU allocated
express	2 hours	Max. 3900	48	1	Up to 4 cores per job

If a job requires a memory per CPU that exceeds the limit per core, that job's count of total CPUs will automatically be increased to fit the memory requirements.

Submit batch jobs

Batch jobs are the most common jobs. Usually, these jobs run applications for long periods without user interaction, so they run autonomously once the resources are available.

These jobs are submitted to the batch system with the ***sbatch*** command:

```
sbatch <slurm-file>
```

where the *slurm-file* has a structure similar to:

```
## SHEBANG #!/bin/bash
## #SBATCH directives
## Load required modules (module load <module>)
## Execute the program (srun <program>)
```

As a summary, below there is a quick reference table with the most common directives used in slurm-files scripts.

Directive	Description
`#SBATCH`	**Required**. Scheduler directive.
`-J, --job-name=<job_name>`	Name of the job that will appear when querying jobs.
`-o, --output=<output_file>`	Defines the name of the file where stdout is redirected.
`-e, --error=<error_file>`	Defines the name of the file where stderr is redirected.
`-p, --partition=<queue_name>`	Submits the job to the specified partition (more info).
`--mail-user=<email>`	Notify user by email when certain event types occur. Events can be: NONE, BEGIN, END, FAIL and REQUEUE.
`--mail-type=<type>`	
`--mem=<size>`	Memory (in MB) required per node.
`--mem-per-cpu=<size>`	Memory (in MB) required per core.
`-n, --ntasks=<num>`	Number of total tasks.
`-c, --cpus-per-task=<num>`	Number of CPU per task (threads).
`--tasks-per-node=<num>`	Tasks per node.
`--gres=gpu:<num_gpus>`	Allocates the indicate number of GPGPUs (1 or 2) per node.
`-t, --time=<dd-hh:mm>`	**Required**. Set a run-time limit for the job allocation. Format: *dd*=days; *hh*=hours; *mm*=minutes

By default, the standard output and standard error are directed to a file named ***slurm-%j.out***, where *%j* is replaced with the job allocation number. This filename can be changed using the *-o* and *-e* options described above. The following replacement pattern options are available:

Filename pattern	Description
%%	The character %.
%A	Job array's master job ID.
%a	Job array ID.
%j	Job ID.
%N	Hostname (will create a separate file per node).
%u	Username.
%x	Job name.

Submit interactive jobs

Interactive jobs allow users to interact in real-time with applications and work directly on a compute node for debugging, using a GUI interface or compiling.

To request resources for an interactive job, execute ***salloc*** command:

```
salloc -n <#tasks> -t <time_limit>
```

and then ***srun*** to start the tasks:

```
srun -n <#tasks> <program>
```

For example, if you need to compile your own code, you only need to start an interactive job where you want to run your application (architecture) and then, set the environment before compiling:

- For standard nodes:

```
salloc -t <time_limit>
```

- For GPU nodes:

```
salloc -p gpu --gres=gpu:1 -t <time_limit>
```

- For KNL nodes:

```
salloc -p knl -t <time_limit>
```

Take a look to the **Modules section** to find a short description of the usage and structure of the Tcl modules system at CSUC.

Monitoring and managing jobs

Some useful commands and options for monitoring and managing jobs are:

- **squeue**: Shows you a list of all running and pending jobs.

  -u <*user*> Show your own running and pending jobs.

  -A Show all jobs by members of your group.

- **scancel**: Cancels a running or pending job into the queue.

  -j <*job_id*> Cancel specified job.

  -u <*user*> Cancel all the jobs of the user (only applicable with your own user).

- **sinfo**: Shows you information about the state of the nodes and partitions.

  -s Report state summary only.

  -p <*partition*> Report on specific partition

- **sacct**: Shows you information about your finished, running or pending jobs.

  -S <*yyyy-mm-dd*> Select jobs eligible after this time. Default is the 00:00 of today.