HPC CHEAT SHEET

CSUC Consorci de Serveis Universitaris de Catalunya

• Logging in: `ssh -p 2122 <username>@hpc.csuc.cat`

Connect with X11 forwarding:

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

• Transferring files:

Upload file to cluster:

```
scp -P 2122 <origin_path> <username>@hpc.csuc.cat:<destiny_path>
```

Download file from cluster:

```
scp -P 2122 <username>@hpc.csuc.cat:<origin_path> <destiny_path>
```

• Storage:

	Variable	Avail.	Quota	Backup	Time limit
/home/$USER	$HOME	Global	25 - 200 GB*	✓	Unlimited
/scratch/$USER	-	Global	1 TB	✗	30 days
/scratch/$USER/tmp/$JOBID	$SHAREDSCRATCH**				7 days
/tmp/$USER/$JOBID	$LOCALSCRATCH**	Local to node	Job file limit	✗	Job time

* The quota is group-wide and depends on the basic pack acquired.

** During job execution, variable $SCRATCH resolves to $SHAREDSCRATCH for multi-node jobs or $LOCALSCRATCH for one-node jobs.

• Partitions (queues):

	Max time	Nodes	Cores / Node	Threads / Core	Memory per core
std (default)	∞	77	48	1	≤ 3900 MB
std-fat	∞	6	48	1	> 3900 MB & ≤ 7900 MB
mem	∞	2	192	1	> 7900 MB & ≤ 24180 MB
gpu*	∞	3	48 + 2 GPGPU	1	≤ 3900 MB
knl	∞	4	68	4	≤ 6 GB

* It is necessary to ask for 24 cores per GPGPU (24 cores / 1 GPGPU or 48 cores / 2 GPGPU).

• Modules

Modules are classified as *apps* (applications), *libs* (libraries), *tools* (compilers and others) and *toolschains* (a group of modules for work with).

Show available modules:

```
module av
```

Load a module:

```
module load <class>/<name>/<version>
   E.g.: module load apps/gaussian/g16b1
```

Switch between modules:

```
module switch <class>/<name>/<version_1> <type>/<name>/<version_2>
   E.g.: module switch apps/gaussian/g16b1 apps/gaussian/g09e1
```

Unload a module:

```
module unload <class>/<name>/<version>
   E.g.: module unload apps/gaussian/g09e1
```

List all loaded modules:

```
module list
```

Unload all loaded modules:

```
module purge
```

• Useful SLURM commands:

Send a batch job:

```
sbatch <slurm_scrip.slm>
```

Useful parameters: `-J <job_name>; -o <out_filename_pattern>*; -e <err_filename_pattern>*; -p <partition>; -w <node_name>; -N <#Nodes>; -n <#tasks>; -c <#threads>; -t <time_limit>; --mem=<memory_per_node>MB; --mem-per-cpu=<memory_per_node>MB`

* Sbatch allows for a filename pattern to contain one or more replacement symbols:

`%j`: Jobid of the running job. `%a`: Job array ID (index) number.

`%u`: Username. `%x`: Job name.

Start an interactive job:

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

Start a task inside a job:

```
srun <program>
```

View state of pending / running jobs:

```
squeue
```

View state of pending / running / finished jobs:

```
sacct -j <job_id>
```

Hold/Release pending jobs:

```
scontrol hold <job_id> / scontrol release <job_id>
```

Cancel a job:

```
scancel <job_id>
```

• SLURM job submission examples:

Standard Serial job:

```
#!/bin/bash                          # Bash directive
#SBATCH -J job_name                  # Job name identification
#SBATCH -e error_file.%j.err         # Path to stderr file
#SBATCH -o output_file.%j.out        # Path to stout file
#SBATCH --mail-user=user@mail.com    # Send an email if the job
#SBATCH --mail-type=BEGIN,END,FAIL   #+ starts / ends / fails
#SBATCH -p std                       # Submit job to queue std
#SBATCH -t 1-00:00                   # Time limit in dd-hh:mm
<Load modules and run application>
```

Standard Parallel MPI job (Generally for multi-node parallelization):

```
#!/bin/bash
#SBATCH -J job_name
#SBATCH -n 4                        # MPI tasks
#SBATCH -t 1-00:00
<Load modules and run application>
```

Standard Parallel OMP job (Generally for intra-node parallelization):

```
#!/bin/bash
#SBATCH -J job_name
#SBATCH -n 1
#SBATCH -c 2                        # OMP threads
#SBATCH -t 1-00:00
<Load modules>
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
<run application>
```

Standard Parallel MPI+OMP job (4x2 cores):

```
#!/bin/bash
#SBATCH -J job_name
#SBATCH --ntasks=4                  # MPI tasks
#SBATCH --cpus-per-task=2           # OMP threads
#SBATCH -t 1-00:00
<Load modules>
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
<run application>
```

Standard job with memory requirement between 3,8GB and 7,8GB per core:

```
#!/bin/bash
#SBATCH -J job_name
#SBATCH -p std-fat                  # Submit job to queue std-fat
#SBATCH -t 1-00:00
<Load modules and run application>
```

High memory demanding job (between 7,8GB and 24GB per core):

```
#!/bin/bash
#SBATCH -J job_name
#SBATCH -p mem                      # Submit job to queue mem
#SBATCH --cpus-per-task=24
#SBATCH -t 1-00:00
<Load modules and run application>
```

GPGPU job:

```
#!/bin/bash
#SBATCH -J job_name
#SBATCH -p gpu                      # Submit job to queue gpu
#SBATCH -gres=gpu:2                 # Request 2 GPGPU P100 per node
#SBATCH -N 1                        # Number of nodes
<Load modules and run application>
```

• SLURM's job environment variables:

SLURM sets some useful environment variables at job start:

SLURM_JOB_NAME	Name of the job.
SLURM_JOB_ID	The ID of the job allocation.
SLURM_ARRAY_JOB_ID	Job array's master job ID number.
SLURM_ARRAY_TASK_ID	Job array's current ID (index) number.
SLURM_ARRAY_TASK_MIN	Job array's minimum ID (index) number.
SLURM_ARRAY_TASK_MAX	Job array's maximum ID (index) number.
SLURM_CPUS_PER_TASK	Number of cpus requested per task.
SLURM_NTASKS	Same as -n, --ntasks.
SLURM_CPUS_PER_TASK	Only set if the --cpus-per-task option is specified.
SLURM_SUBMIT_DIR	The directory from which sbatch was invoked.

• SLURM considerations:

SLURM does not accept spaces between *#* and *SBATCH* word:

```
# SBATCH -J job_name      ✕ INCORRECT
#SBATCH -J job_name       ✓ CORRECT
```

The following options can be omitted:

-c, --cpus-per-task	Default value = 1
-p, --partition	Default value = std
--mem-per-cpu	Default value = depends on selected partition

The following option is always required:

-t, --time