Introduction to MARCONI

Fabio Affinito
European Infrastructural Projects Team
CINECA

MARCONI

Compute Nodes: 1512 36-core compute nodes for A1, 3600 68-core compute nodes for A2.

– The nodes have 128GB of memory, but the allocatable memory on the node is 118 GB for A1, and 96 GB for A2

– Not all nodes are available for all the users. A partition of the cluster is reserved for EUROFusion community, and the rest is available for academical users

- **Login nodes**: 8 Login (3 available for regular users) & 12 service nodes for cluster management, each one contains 2 x Intel Xeon Processor E5-2697 v4 with a clock of 2.30GHz and 128 GB of memory. The login nodes are shared between A1 and A2, while the service nodes are splitten among the partitions (6 for A1 and 6 for A2).

Login

In order to connect to the MARCONI login nodes:

ssh <username>@login.marconi.cineca.it

You can use the terminal on your linux distribution (or MacOS X) or you can use a client ssh from Windows (e.g. PuTTY or MobaXterm).

There is also a plugin on Google Chrome to emulate a terminal.

Don't use the login node for calculations!

Storage

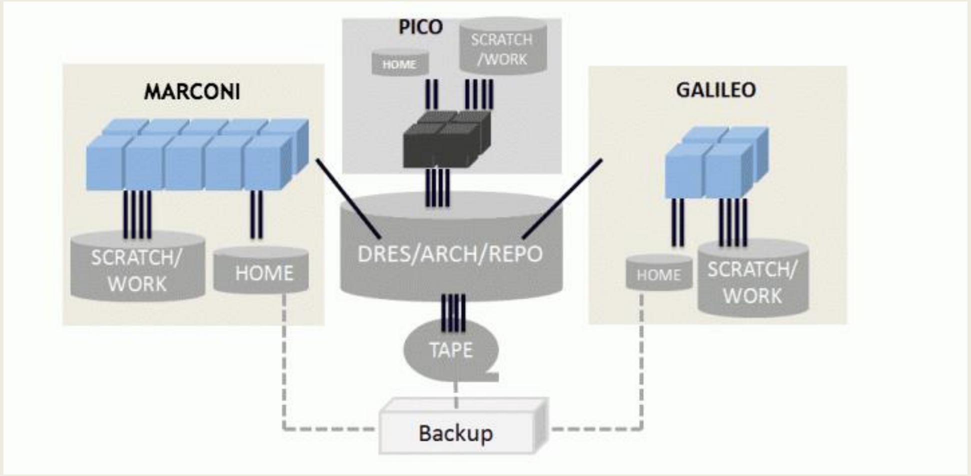
The storage of the HPC infrastructure is organized in different areas:

$HOME

$CINECA_SCRATCH

$WORK

Plus a long term storage



Storage

$HOME: Permanent, backed-up, and local to MARCONI. 50 Gb of quota. For source code or important input files.

$CINECA_SCRATCH: Large, parallel filesystem (GPFS). No quota. Run your simulations and calculations here.

A cleaning procedure automatically deletes every file untouched since 50 days

$WORK: Similar to $CINECA_SCRATCH, but the content is shared among all the users of the same account. 1 Tb of quota.

Module system

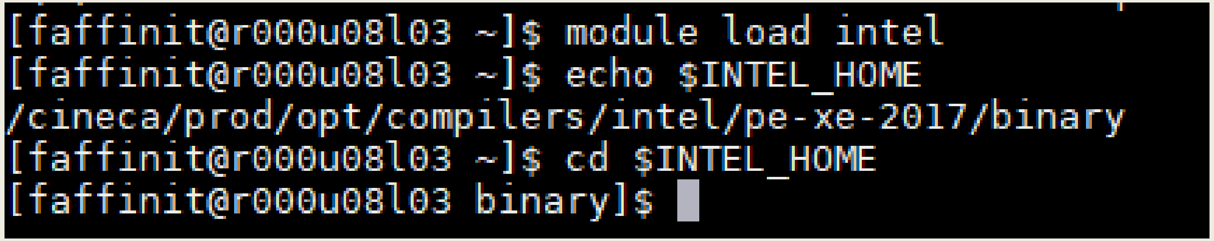
All the optional software on the system is made available through the "module" system. It provides a way to rationalize software and its environment variables.

Modules are divided in several profiles:

- **profile/base default** - stable and tested compilers, libraries, tools
- **profile/advanced** libraries and tools compiled with different setups that the default
- **profile/chem(/phys, bioinf, astro,...)** “domain” profiles with the application softwares specific for each field of research
- **profile/archive** old or outdated versions of our module (we don't throw away anything!)

Module system

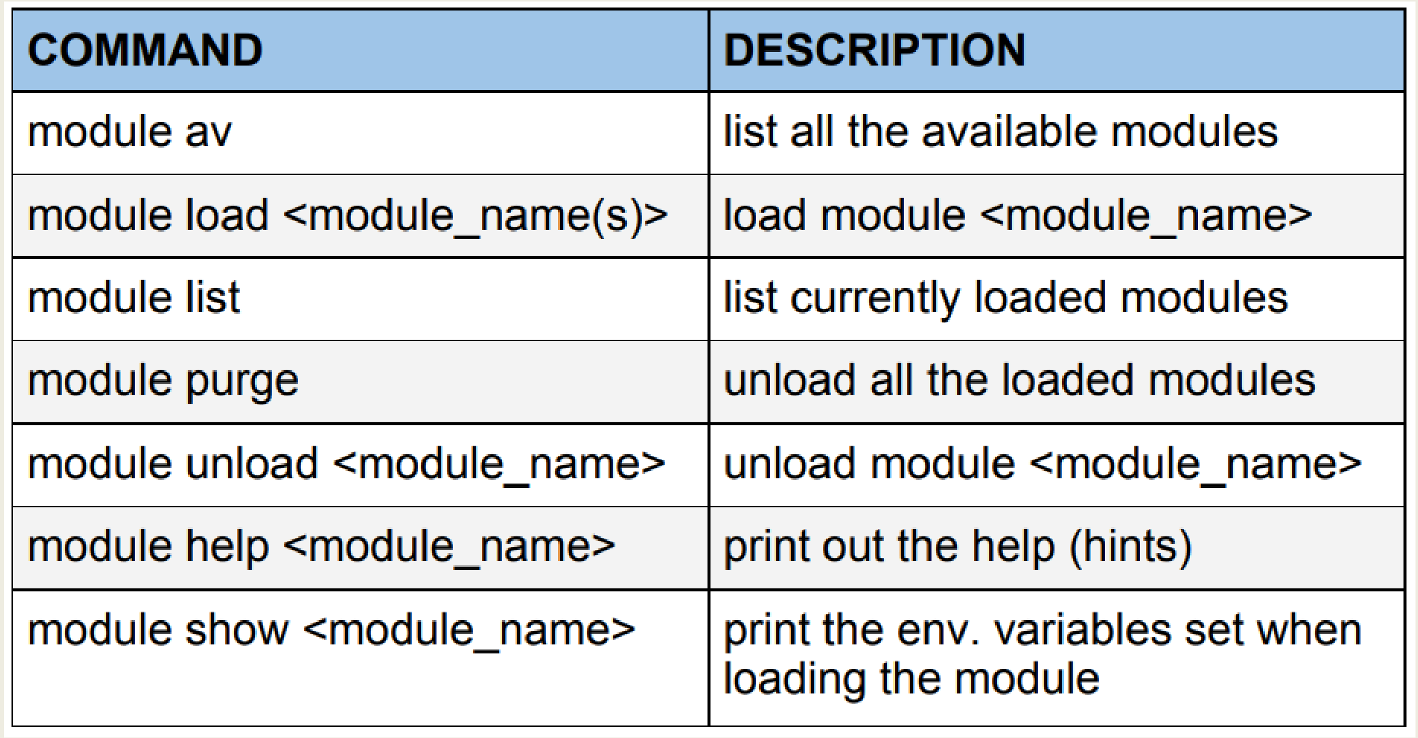
When a module is loaded, a set of environment variables is set.



```
[faffinit@r000u08l03 ~]$ module load intel
[faffinit@r000u08l03 ~]$ echo $INTEL_HOME
/cineca/prod/opt/compilers/intel/pe-xe-2017/binary
[faffinit@r000u08l03 ~]$ cd $INTEL_HOME
[faffinit@r000u08l03 binary]$
```

For certain modules, a specific profile must be loaded before ("module load profile/..."). Use the "modmap" command to understand which module is in which profile (try "modmap -h")

Module commands



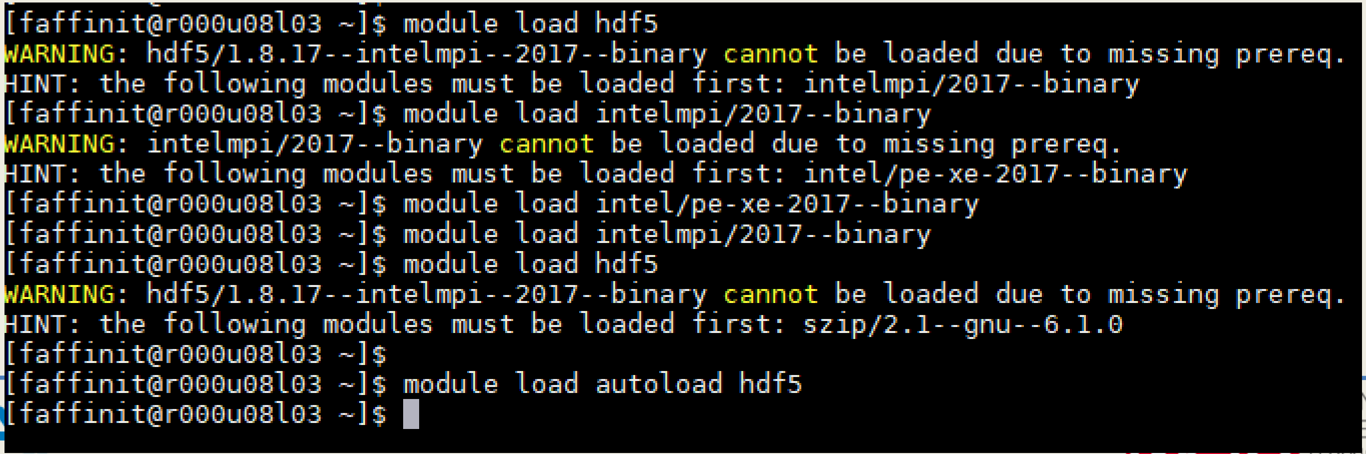
COMMAND	DESCRIPTION
module av	list all the available modules
module load <module_name(s)>	load module <module_name>
module list	list currently loaded modules
module purge	unload all the loaded modules
module unload <module_name>	unload module <module_name>
module help <module_name>	print out the help (hints)
module show <module_name>	print the env. variables set when loading the module

SuperComputing Applications and Innovation

DRIVING THE EXASCALE TRANSITION

Module dependencies

Some modules need to be loaded after other modules they depend from (e.g.: parallel compiler depends from basic compiler). You can load both compilers at the same time with "autoload".



```
[faffinit@r000u08l03 ~]$ module load hdf5
WARNING: hdf5/1.8.17--intelmpi--2017--binary cannot be loaded due to missing prereq.
HINT: the following modules must be loaded first: intelmpi/2017--binary
[faffinit@r000u08l03 ~]$ module load intelmpi/2017--binary
WARNING: intelmpi/2017--binary cannot be loaded due to missing prereq.
HINT: the following modules must be loaded first: intel/pe-xe-2017--binary
[faffinit@r000u08l03 ~]$ module load intel/pe-xe-2017--binary
[faffinit@r000u08l03 ~]$ module load intelmpi/2017--binary
[faffinit@r000u08l03 ~]$ module load hdf5
WARNING: hdf5/1.8.17--intelmpi--2017--binary cannot be loaded due to missing prereq.
HINT: the following modules must be loaded first: szip/2.1--gnu--6.1.0
[faffinit@r000u08l03 ~]$
[faffinit@r000u08l03 ~]$ module load autoload hdf5
[faffinit@r000u08l03 ~]$
```

Module conflicts

You may also get a “conflict error” if you load a module not suited for working together with other modules you already loaded (e.g. different compilers). Unload the previous module with “module unload”

Compilers

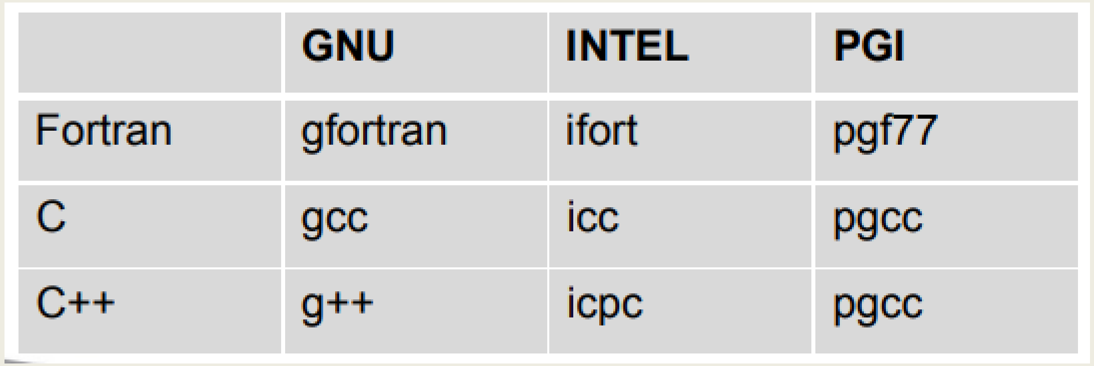
- On MARCONI you can choose between three different compiler families: gnu, intel and pgi
- You can take a look at the versions available with “module av” and then load the module you want.

module load intel loads default intel compilers suite

module load intel/pe-xe-2017--binary loads specific compilers suite

Compilers

Compilers available on MARCONI



	GNU	INTEL	PGI
Fortran	gfortran	ifort	pgf77
C	gcc	icc	pgcc
C++	g++	icpc	pgcc

Parallel libraries

To compile a parallel MPI program, you need to link the MPI library. This is implicit when you use a compiler wrapper.

On MARCONI, you can choose to use IntelMPI or OpenMPI.

Before loading a MPI library, you need to load an appropriate compiler, in advance.

```
module av openmpi
 openmpi/1-10.3--gnu--6.1.0 (profile/base)
 openmpi/1-10.3--intel--pe-xe-2017--binary
 (profile/advanced)
module load autoload openmpi/1-10.3--gnu--6.1.0
```

Parallel libraries

Name of the MPI wrappers.

	OPENMPI/INTELMPI
Fortran90	mpif90/mpiifort
C	mpicc/mpiicc
C++	mpiCC/mpiicpc

Flags are the same used with the standard compilers.

For example, for OpenMP you will use –fopenmp for the GNU compiler or –qopenmp for the Intel compiler.

Introduction to SLURM

In any parallel cluster, the management of the resources is enforced by a scheduler.

The scheduler manages the requests of the users and provides the resources trying to guarantee a fair sharing.

On MARCONI, the resource manager is SLURM.

You can ask for an interactive session or you can submit a job script for the execution of your job.

Interactive job

```
srun    -N<nodes_no>
        --ntasks-per-node=<tasks_per_node_no>
        -A<account_no>
        --partition=<name>  (--time ... )
        --pty bash
```

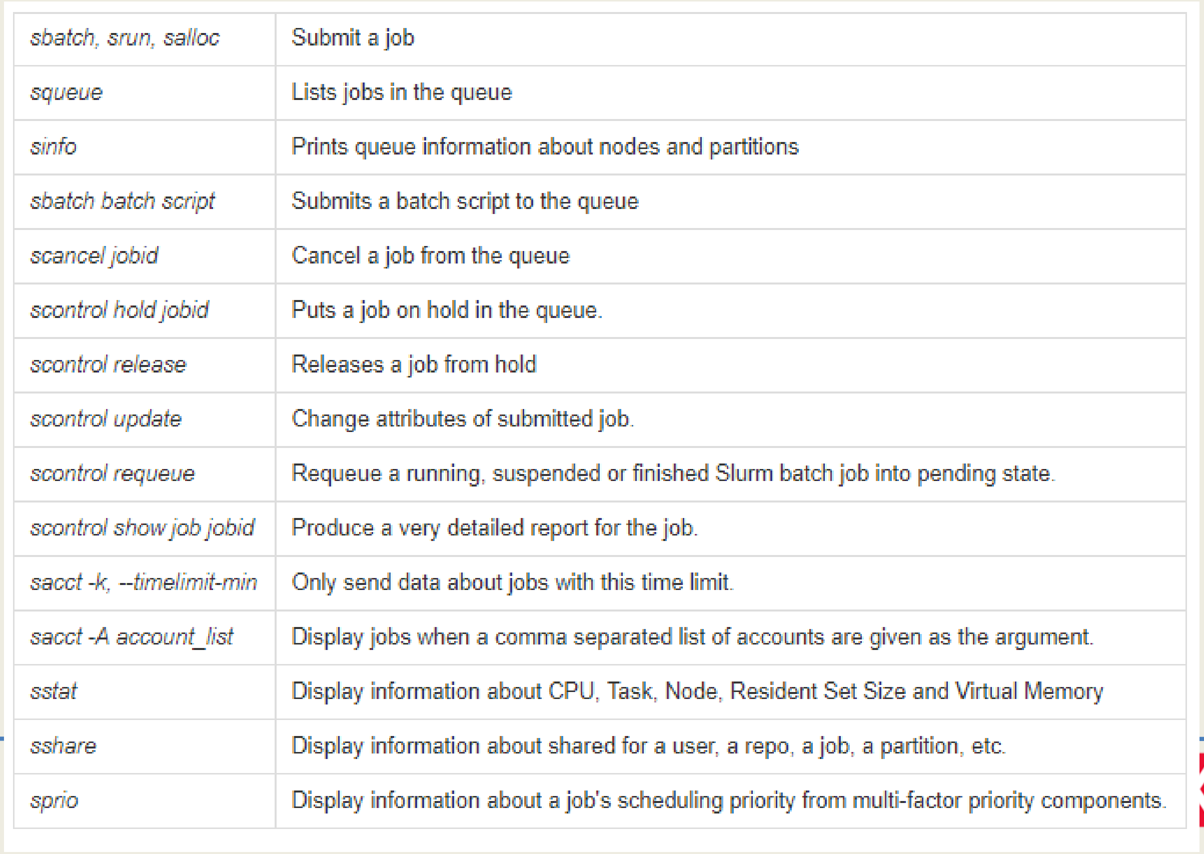
When this command is executed, your request is inserted in a queue. When the required resources are ready, you will be returned with a bash shell on the compute nodes.

Batch job

```
#!/bin/bash
#SBATCH -N1 --ntasks-per-node=36     # 36 cores on 1 node
#SBATCH --time=1:00:00                # time limits: 1 hour
#SBATCH --error=myJob.err             # standard error file
#SBATCH --output=myJob.out           # standard output file
#SBATCH --account=<account_no>       # account name
#SBATCH --partition=<partition_name> # partition name
#SBATCH --qos=<qos_name>             # quality of service
srun …
```

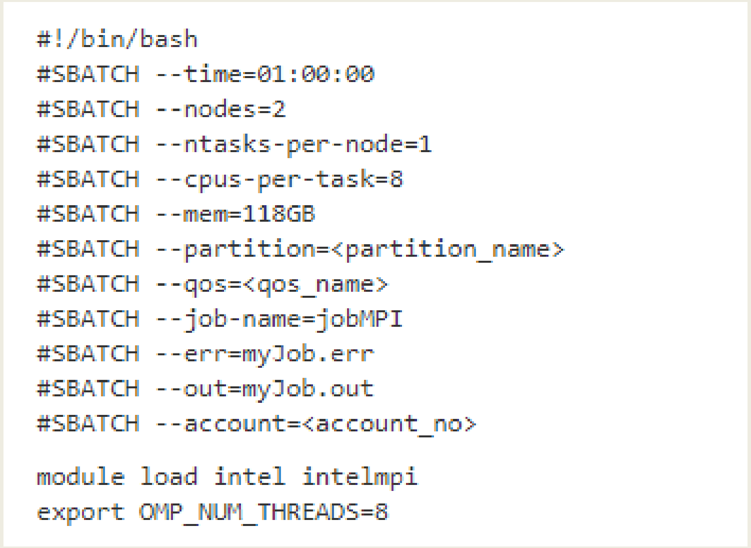
When you submit this job script, the resource manager takes in charge its execution that will happen when the requested resources will be ready

Basic SLURM commands



sbatch, srun, salloc	Submit a job
squeue	Lists jobs in the queue
sinfo	Prints queue information about nodes and partitions
sbatch batch script	Submits a batch script to the queue
scancel jobid	Cancel a job from the queue
scontrol hold jobid	Puts a job on hold in the queue.
scontrol release	Releases a job from hold
scontrol update	Change attributes of submitted job.
scontrol requeue	Requeue a running, suspended or finished Slurm batch job into pending state.
scontrol show job jobid	Produce a very detailed report for the job.
sacct -k, --timelimit-min	Only send data about jobs with this time limit.
sacct -A account_list	Display jobs when a comma separated list of accounts are given as the argument.
sstat	Display information about CPU, Task, Node, Resident Set Size and Virtual Memory
sshare	Display information about shared for a user, a repo, a job, a partition, etc.
sprio	Display information about a job's scheduling priority from multi-factor priority components.

A template for the execution



```
#!/bin/bash
#SBATCH --time=01:00:00
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=8
#SBATCH --mem=118GB
#SBATCH --partition=<partition_name>
#SBATCH --qos=<qos_name>
#SBATCH --job-name=jobMPI
#SBATCH --err=myJob.err
#SBATCH --out=myJob.out
#SBATCH --account=<account_no>

module load intel intelmpi
export OMP_NUM_THREADS=8
```

Parallel (MPI) programs can be runned using:

> mpirun –n <procs> ./myprogram

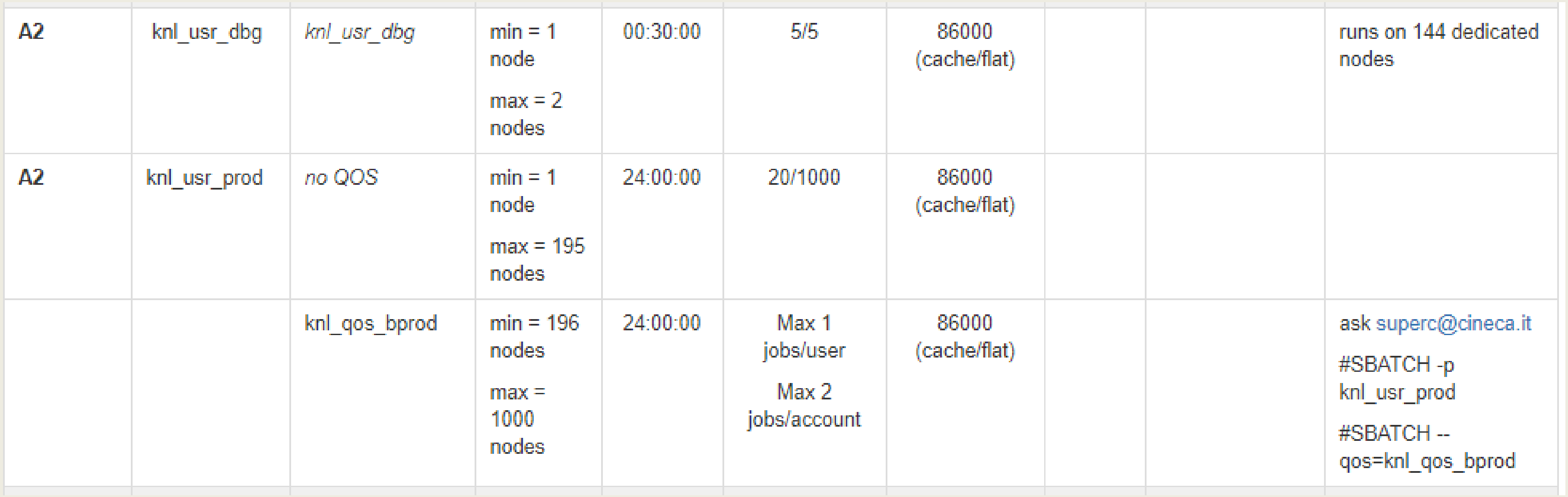
or

> srun --mpi=pmi2 –n <procs> ./myprogram

MARCONI A1 Partitions

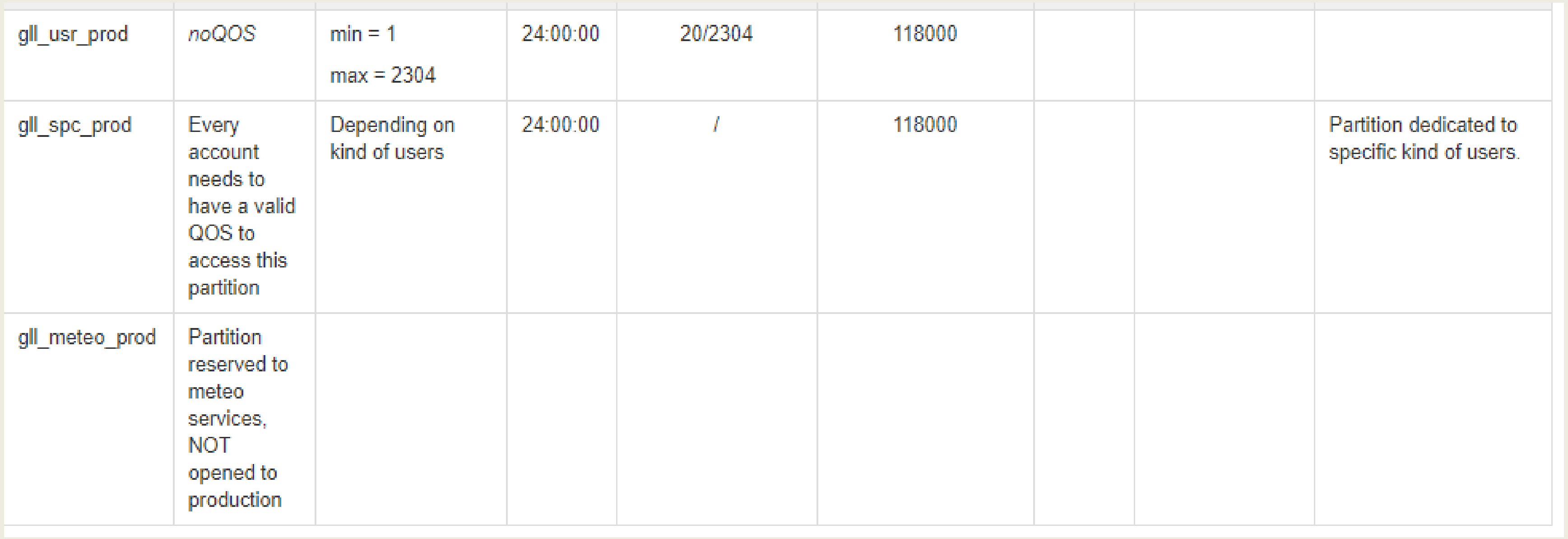
A1	bdw_usr_dbg	*noQOS*	min = 1 max = 144	00:30:00	4/144	118000			managed by route runs on 24 nodes shared with the visualrcm queue
A1	bdw_usr_prod	*noQOS*	min = 1 max = 2304	24:00:00	20/2304	118000			
		bdw_qos_bprod	min = 2305 max = 6000	24:00:00	1/6000	118000			#SBATCH -p bdw_usr_prod #SBATCH --qos=bdw_qos_bprod
		bdw_qos_special	min = 1 max = 36	180:00:00		118000			ask superc@cineca.it #SBATCH -p bdw_usr_prod #SBATCH --qos=bdw_qos_special

MARCONI A2 Partitions



A2	knl_usr_dbg	*knl_usr_dbg*	min = 1 node max = 2 nodes	00:30:00	5/5	86000 (cache/flat)			runs on 144 dedicated nodes
A2	knl_usr_prod	*no QOS*	min = 1 node max = 195 nodes	24:00:00	20/1000	86000 (cache/flat)			
		knl_qos_bprod	min = 196 nodes max = 1000 nodes	24:00:00	Max 1 jobs/user Max 2 jobs/account	86000 (cache/flat)			ask superc@cineca.it #SBATCH -p knl_usr_prod #SBATCH --qos=knl_qos_bprod

GALILEO Partitions



gll_usr_prod	*noQOS*	min = 1 max = 2304	24:00:00	20/2304	118000				
gll_spc_prod	Every account needs to have a valid QOS to access this partition	Depending on kind of users	24:00:00	/	118000				Partition dedicated to specific kind of users.
gll_meteo_prod	Partition reserved to meteo services, NOT opened to production								