UG2.6: Production Environment

In this page:

Additional page:

SLURM Scheduler

Since our HPC systems are shared among many users, long production jobs should be submitted using a scheduler. This guarantees that access to our resources is as fair as possible.

Roughly, there are two different modes to use an HPC system:

- **interactive**, for data movement, archiving, code development, compilations, basic debugger usage: also for very short test runs and general interactive operations. A task in this class should not exceed 10 minutes CPU-time and is free of charge on HPC systems with the current billing policy.
- **batch**, for production runs. Users must prepare a shell script containing all the operations to be executed in batch mode, once the requested resources are available and assigned to the job. The job then starts and executes on compute nodes of the cluster. Remember to put all your data, programs and scripts in the $WORK or $CINECA_SCRATCH filesystem, which are the best storage areas accessible by compute nodes.

  You must have valid active projects on the system in order to run batch jobs. Moreover, remember that on our systems there may be specific policies for the use of project budgets.

On all of our current HPC systems, the queuing system or scheduler is SLURM. **SLURM Workload Manager (or simply SLURM, which stands for "Simple Linux Utility for Resource Management")** is an open-source and highly scalable job scheduling system.

SLURM has three key functions. Firstly, it allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time, so they can perform their work. Secondly, it provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes. Finally, it arbitrates contention for resources by managing the queue of pending jobs.

Comprehensive documentation of SLURM and some **examples on how to submit your job** is described in a separate section under this chapter, as well as on the original SchedMD site

The Software Catalog

CINECA offers a variety of third-party applications and community codes that are installed on its HPC systems. Most of the third-party software is installed using software modules mechanism (see "The module command" later in this section).

Information on the available packages and their detailed descriptions are organized in a full catalog optionally divided by discipline in our web site by selecting "*software*" and "*Application Software for Science*". It can be also obtained working interactively on the HPC systems, using the module or modmap commands (see later in this chapter).

If you do not find an application you are interested in on our website or on the specific system or if you have a question about currently available software, please contact our specialists (superc@cineca.it).

The "module" command

All software programs installed on the CINECA machines are available as modules.

A basic default modules environment is already set up by the system login configuration files.

In order to have a list of available modules and select a specific one, you have to use the **module** command. The following table contains its basic options:

```
Command               Action
--------------------------------------------------------------------------------------------
module avail .................. show the available modules on the machine
module load <appl> ............ load the module <appl> in the current shell session, preparing the environment
for the application.
module load autoload <appl> ... load the module <appl> and all dependencies in the current shell session
module help <appl> ............ show specific information and basic help on the application
module list ................... show the modules currently loaded on the shell session
module purge .................. unload all the loaded modules
module unload <app>............ unload a  specific module
--------------------------------------------------------------------------------------------
```

As you will see by typing "module avail", the software modules are collected in **different profiles** (base, advanced....) and organized by **functional categories** (compilers, libraries, tools, applications,..).

In order to **detect all profiles, categories, and modules available** on our systems the command “modmap” is available.

$ modmap -m <namesoftware>

It shows information about all available versions installed of the software of interest, and in which profile you can find them.

We strongly recommend loading a module with its full name, i.e. the name of the module with its version (you can find the version with *modmap* or *module av*). Without the full name, we cannot guarantee that it will work. As an example: on G100, the full-name for python 3.8.12 compiled with gcc 10.2.0 is: python/3.8.12–gcc–10.2.0

ATTENTION: Remember to load the needed modules in batch scripts too, before using the related applications.

How to install your software with Spack

In case you don't find a software, you can choose to install it by yourself.
In this case, we also offer the possibility to use the “spack” environment provided by the package manager Spack. In the following, we introduce the essential commands required for the installation of a software package.

To start using Spack on Marconi100 or Galileo100, load the corresponding module:

```
$ module load spack
```

If you want to install a software package, check before if it is available via spack by typing the following spack command

```
$ spack list <package_name>
```

and also check if it is already installed with

```
$ spack find -ldvp <package_name>
```

where the suggested options show respectively the hash, the dependencies, the variants and the installation path.

In order to install a package with the spack module you have to specify for it a version, a compiler, the dependencies and the building variants or to use the default options. The combination of all these parameters is the "spec" with which the package will be installed. Before installing a software package, run

```
$ spack spec -Il <package_name>
```

where flags "-Il" show the installation status and the hash.

Then you can simply install your package via spack with the default options shown by the spec command

```
$ spack install <package_name>
```

or you can specify, for example, the compiler and the dependency from a library , e.g. gcc 8.3.0 and zlib 1.2.11

```
$ spack install <package_name> %gcc@8.3 ^zlib@1.2.11
```

You can use <package_name> or /<package_hash> without distinction, here for simplicity all the examples are reported with <package_name>.

In order to use the software just installed via spack, you can load it with the spack command

```
$ spack load <package_name>
```

Please refer to UG2.6.2: How to install via Spack on Cineca clusters and to the official documentation of Spack for more detailed instructions and more examples on the use of spack commands.

Python and additional software

In case you need a particular Python package that is not already installed on our systems you can install it by yourself, defining your python virtual environment.

You can follow these instructions:

- load python interpreter from the module

```
$ module load python/3.8.2
```

- create a virtual environment which is essentially a new directory (my_venv) containing all you need

```
$ python3 -m venv my_venv
```

- activate the new virtual env

```
$ source my_venv/bin/activate
```

- install whatever you need (e.g matplotlib)

```
(my_venv) $ pip3 install matplotlib
```

- when you have finished your business, you can deactivate the virtualenv by

```
(my_venv) $ deactivate
```

Some packages (numpy, scipy, ...) could be already available as modules. Check with

```
$ modmap -m <package_name>
```

Deep Learning domain: the CINECA Artificial Intelligence Project and additional software

The bulk of the cineca-ai package, provided by the deeplrn profile, is based on the Open Cognitive Environment (Open-CE) tool, which includes (for example) Tensorflow, Pytorch, XGBoost, and other related packages and dependencies. This cognitive environment has been personalised by CINECA AI experts and published in a public channel.

Several versions of the package are available in profile/deeplrn, differing in the versions of the contained packages. To see what is available:

```
$ modmap -m cineca-ai
```

The module help reports the versions of the main components of the package for each module and the basic instructions to use it:

```
$ module load profile/deeplrn
$ module av cineca-ai
$ module help cineca-ai/<version>
```

You can rely on the rich catalog offered by the CINECA-AI channel and build your python/conda environment on top of it to install additional software following the How-To guide.