


# Best practice guidelines/tutorials for MSA/heterogeneous systems

# Table of Contents

# List of Figures

# List of Tables

## Executive Summary

During the project, different high-performance computing systems will be made available to the partners to develop, test, and produce the various software elements implemented in the CoE RAISE. This report delivers a best practice guideline and tutorials for the usage of the modular and heterogeneous systems available at Forschungszentrum Jülich, University of Iceland, RWTH Aachen University, Barcelona Supercomputing Center, and Riga Technical University. In contrast, the “Best practice guidelines / tutorials prototype” document of RAISE focuses on the description of prototype machines to enable the early porting of RAISE’s software stacks to future architectures.

The modular and heterogeneous systems presented in this report will be critical for implementing AI tools developed in work package 2 (WP2, AI- and HPC-Cross Methods at Exascale) of RAISE and their integration in the different applications of the work package 3 (WP3, Compute-Driven Use-Cases at Exascale) and work package 4 (WP4, Data-Driven Use-Cases at Exascale). The selected systems offer sufficient modularity and a great variety of architectures to satisfy the requirements of these applications. They will be used to implement AI tools, integrate these tools in application software, validate the proposed strategies, and to demonstrate their applicability through the solution of the challenges proposed in WP3 and WP4.

If the resources offered by the different systems are not sufficient to carry out a particular challenge, the partners will use internal resources or submit proposals to national access calls or EU Partnership for Advanced Computing in Europe (PRACE) access calls. This includes future calls for resources handled under the umbrella of the EuroHPC Joint Undertaking, possibly managed by PRACE.

# 1 Introduction

This report provides a self-contained description of the different supercomputers made available to the partners of the project and which are accessible for the community via open compute-time calls. To provide the reader with additional information, links to the main web pages, which are continuously updated, are given. It should be noted that some of the systems are already in production, while others will become available soon.

The different supercomputing centers in RAISE are well distributed over Europe, see Figure 1. They offer a large variety of architectures to embrace the porting of the project's various applications. The different Central Processing Unit (CPU) architectures represented are IBM Power9, Intel Broadwell, Intel Sandy Bridge, Intel Skylake, Intel Xeon Platinum, Intel Xeon Gold, Intel Xeon Phi (KNL), and AMD EPYC. Many of these systems comprise a dedicated node with accelerators, mainly NVIDIA Graphics Processing Units (GPUs) like Volta and Tesla.

In the following, the High-Performance Computing (HPC) systems are presented consecutively. Section 2 contains information on the CTE-POWER system installed at the Barcelona Supercomputing Center (BSC). In Sec. 3, the Nord3 system at BSC is described. Subsequently, details on the systems JUWELS (Jülich Wizard for European Leadership Science) and JURECA (Jülich Research on Exascale Cluster Architectures), located at the Jülich Supercomputing Centre (JSC), Forschungszentrum Jülich (FZJ), are given. Details on the system RUDENS at Riga Technical University (RTU) can be found in Sec. 5. RWTH Aachen University's system CLAIX is described in Sec. 6. Finally, Sec. 7 provides information on the systems GARPUR, installed at the University of Iceland (UOI) and the upcoming LUMI pre-exascxale system.
Each section starts with an overview of the system hardware configuration. Access to these supercomputers will be granted via a procedure specific to the project's partner, as indicated in the "Accessing the system" sections. The first set of instructions will help new users to login and carry out their first steps. A subsequent user's guide describes the software stacks, module systems, and the available file systems. This is followed by a presentation of how to develop software on the systems and a user's guide of the batch systems.


**Figure 1: Locations and names of the modular and heterogeneous HPC systems provided within RAISE.**

# 2 CTE-POWER at Barcelona Supercomputing Center

The CTE-POWER cluster consists of IBM POWER9 processors and NVIDIA Volta GPUs, which are the same components that IBM and NVIDIA use for the Summit[1] and Sierra[2] supercomputers hosted by the Oak Ridge and Lawrence Livermore National Laboratories. It has a computing power over 1.5 Petaflops per second (PFLOPs).

In the following, more details of the hardware specifications of the CTE-POWER system are given in Sec. 2.1. Section 2.2 describes how to access the system, before Sec. 2.3 provides a user guideline on how to work with the system.

## 2.1 Hardware specifications

The CTE-POWER is a cluster system based on the IBM POWER9 processor technology, with a Linux Operating System and an Infiniband interconnection network.

There exist two login nodes and 52 compute nodes, each of them with the following configuration:

- 2 x IBM Power9 8335-GTH, clocked at 2.4GHz (3.0GHz on turbo, 20 cores and 4 threads/core, in total 160 threads per node)
- 512 GB of main memory distributed over 16 Dual Inline Memory Modules (DIMMs) x 32 GB @ 2666MHz
- 2 x Solid-state disc (SSD) 1.9 TB as local storage
- 2 x 3.2 TB Non-Volatile Memory Express (NVMe)
- 4 x GPU NVIDIA V100 (Volta) with 16 GB High-Bandwidth Memory (HBM) 2.
- Single Port Mellanox Endpoint Detection and Response (EDR) network
- General Parallel File System (GPFS) via a single 10 Gbit/s fiber link

The operating system is a Red Hat Enterprise Linux Server 7.5 alternative.

## 2.2 Accessing the system

Accessing the CTE-POWER system is quite simple. In the following, the method on how to apply for an account is described in Sec. 2.2.1, before explanations on how to login and change the password are given in Sec. 2.2.2.

### *2.2.1 Account application*

The BSC direct collaborators in the project can gain access through the following steps:

1. An email has to be sent to guillaume.houzeaux@bsc.es asking for system access. In case of emergency, the deputy oriol.lehmkuhl@bsc.es should be contacted.
2. Once the petition is accepted, generally in hours or few days, the user's responsibilities agreement needs to be signed electronically.
3. The applicant will then receive an email with all the instructions.

The access will be granted for the whole duration of the project.

[1] Summit https://www.olcf.ornl.gov/summit/
[2] Sierra https://computing.llnl.gov/computers/sierra

### *2.2.2 Login and password management*


**Figure 2: CTE-POWER login screen.**

Two public login nodes are available to connect to the CTE-ARM system. The login nodes are:

- `plogin1.bsc.es`
- `plogin2.bsc.es`

Upon login, the user will be provided with a shell on a login node, where code can be compiled and the applications can be prepared, see Figure 2.

The `ssh` tools need to be used to login into the system and to transfer files to and from the cluster. Incoming connections from protocols like `telnet`, `ftp`, `rlogin`, `rcp`, or `rsh` are not allowed. It should be noted that only incoming connections are allowed on the whole cluster, i.e., once logged in, outgoing connections are rejected for security reasons.

It is recommended to change the password on the first login. To change the password, the user has to login to a different machine (`dt01.bsc.es`). The corresponding connection must be established from the user's local machine.

```
local$       ssh -l username dt01.bsc.es
dttransfer1$ passwd
             Changing password for username.
             Old Password:
             New Password:
             Reenter New Password:
             Password changed.
```

The password change takes about 10 minutes to be effective.

## 2.3 Using the system

In the following, the software stack is first described and an introduction to the module system is given in Sec. 2.3.1. The available file systems and the data transfer mechanisms are introduced in Sec. 2.3.2. Section 2.3.3 explains how to do software developments on the system and Sec. 2.3.4 describes how to use the batch system on the CTE-POWER cluster.

### *2.3.1 Software stack and introduction to the module system*

**Modules Environment**

The `Environment Modules` package[3] provides a dynamic modification of a user's environment via module files. Each module file contains the information needed to configure the shell for an application or a compilation. Modules can be loaded and unloaded dynamically in a clean fashion. All popular shells are supported, including `bash`, `ksh`, `zsh`, `sh`, `csh`, `tcsh`, as well as some scripting languages such as `perl`.

The software packages installed on the CTE-POWER system are divided into five categories:

- **Environment:** This category contains module files dedicated to preparing the environment, e.g., get all necessary variables to use Open Message Passing Interface (OpenMPI) to compile or run programs.
- **Tools:** Contains all useful tools, which can be used at any time (`php`, `perl`, etc.).
- **Applications:** Contains various pre-installed HPC applications (GROMACS, etc.).
- **Libraries:** Contains libraries frequently loaded at compile time (FFTW, LAPACK, etc.). The correct compiler and linker flags are automatically loaded into the environment.
- **Compilers:** Contains various compiler suites available for the system (`Intel`, `GCC`, etc.).

Modules can be invoked in two ways:

- **By their name alone:** Invoking them by name implies loading the default module version, which is usually the most recent version that has been tested to be stable (recommended) or the only version available. The following example loads the Portland Group (PGI) compiler suite module solely by its name:

```
$ module load pgi
```

[3] Environment Modules package http://modules.sourceforge.net

- **By their name and version:** Invoking by version loads a specific application. As of this writing, the previous command and the following one load the same module.

```
$ module load pgi/18.1
```

The most important commands for using the module system are, (cf. also Sec. 4.3.1 on `Easybuild`):

- `module list`: Shows all the loaded modules.
- `module avail`: Shows all the modules the user is able to load.
- `module purge`: Removes all the loaded modules.
- `module load <modulename>`: Loads the necessary environment variables for the selected module file `<modulename>` (`PATH`, `MANPATH`, `LD_LIBRARY_PATH`, etc.).
- `module unload <modulename>`: Removes all environment changes made by the `module load <modulename>` command w.r.t. the module `<modulename>`.
- `module switch <oldmodule> <newmodule>`: Unloads the first module `<oldmodule>` and loads the second module `<newmodule>`.

The command `module help` can be used any time to check the command's usage and additional options. Furthermore, the `module(1) manpage` can be checked for further information.

**Special BSC commands**

The support team at BSC provides some useful commands for the users' awareness and ease of use of BSC's HPC machines. A short summary of these commands follows:

- `bsc_queues`: Shows the queues available to the user and the time/resource limits.
- `bsc_quota`: Shows a comprehensible quota usage summary for all accessible file systems.
- `bsc_load`: Displays job load information across all related computing nodes.

All available commands have a dedicated `manpage` (not all commands are available on all machines). More information can be obtained by:

```
$ man <bsc_command>
```

e.g., using:

```
$ man bsc_quota
```

### *2.3.2 Available file systems and data transfer*

**IMPORTANT:** It is in the user's responsibility to backup all critical data. BSC only guarantees a daily backup of user data under `/gpfs/home`. Any other backup should only be done exceptionally if demanded by the interested user.

Each user has several areas of disk space for storing files. These areas may have size or time limits. Please read all this section carefully to know about the policy of usage of each of these filesystems.

There are 4 different types of storage available on the cluster:

- The **root filesystem** is the filesystem where the operating system resides. There is a separate partition of the local hard drive mounted on `/tmp` that can be used for storing user data.
- The **GPFS** is a distributed networked filesystem, which can be accessed from all the nodes and the data transfer machine. The parallel filesystem GPFS allows parallel applications simultaneous access to a set of files (even a single file) from any node that has the file system mounted while providing a high level of control over all file system operations. An incremental backup is performed daily only for `/gpfs/home`.
- A **local hard drive** is installed in every node, which can be used as a local scratch space to store temporary files during job execution.
- An NVMe local partition is installed in every node.

The following Table 1 holds information on the available file systems on CTE-POWER.

| Mount point | Description |
|---|---|
| `/apps` | On this file system, applications and libraries that have already been installed on the machine reside. It is recommended to take a look at the directories to know the applications available for general use. |
| `/home` | This file system holds the home directories of all users. Each user has their own home directory to store developed sources and their personal data. A default quota will be enforced on all users to limit the amount of data stored there. **Note:** Jobs should not be run from this file system. Instead, jobs should be executed from the group's `/gpfs/projects` or `/gpfs/scratch` folders. |
| `/gpfs/projects` | For each group of users, there is a directory in `/gpfs/projects`, e.g., the group `bsc01` has the directory `/gpfs/projects/bsc01` ready to use. The space is intended to store data that needs to be shared between the users of the same group or project. A quota per group will be enforced depending on the space assigned by BSC's Access Committee. It is in the project manager's responsibility to |

| Mount point | Description |
|---|---|
| | determine and coordinate the best use of this space, and to define policies how to distribute or shared it between the users. |
| `/gpfs/scratch` | Each user has a directory on `/gpfs/scratch`. Its intended use is to store temporary files of the jobs during their execution. A quota per group is enforced depending on the space assigned. |
| `/scratch/tmp/$JOBID` | This space is local to the nodes. The parameter `$JOBID` corresponds to the id of the job. All data stored on the local hard drives of the compute nodes is not available from the login nodes. Node-local, it is available via the `$TMPDIR` environment variable. The amount of space within the /scratch filesystem is about 1.6 TB. **NOTE:** This directory is for temporary files created during job executions. This directory is automatically cleaned after the job finishes. |
| `/nvme1`<br>`/nvme2` | Every node has two NVMe drives (3 TB each) that can be used as a local working directory to speed-up the code by reducing notably the disk's access time. They can be used with the same methodology as the one described for the local hard drives. There are environment variables available to refer to the directories mounted on each drive, i.e., `$NVME1DIR` and `$NVME2DIR`. |

**Table 1: Available file systems on CTE-POWER.**

### Quotas

The quotas are the amount of storage available for a user or a groups' users. It can be pictured as a small disk readily available to the user. A default value is applied to all users and groups and cannot be outgrown.

The quota can be inspected at any time by using the following command from inside each filesystem (see also Sec. 2.3.1):

```
$ bsc_quota
```

The command provides a readable output for the quota.

If more disk space on any file system of CTE-POWER is required, the principal investigator (PI) of the project has to make a request for the extra space needed, specifying the requested space and the reasons why it is needed. For more information BSC's support (support@bsc.es) can be contacted.

### Transferring files

There are two ways to copy files from/to the cluster:

- direct `scp` or `sftp` to the login nodes
- using a data transfer machine, which shares all the GPFS filesystem for transferring large files

### Direct copy to the login nodes

No connections are allowed from inside the cluster to the outside world. That is, all `scp` and `sftp` commands have to be executed from the user's local machine. Examples on how to use these commands are given subsequently.

On a Windows-based system, most of the `ssh` clients such as `Putty` come with a tool to securely copy files via `scp` or `sftp`. For details on how to use these tools, the reader is referred to the corresponding `ssh` client's manual.

### Data transfer machine

BSC provides special machines for transferring large amounts of data. These machines are dedicated to the data transfer and are accessible through `ssh` with the same account credentials as used to login to the cluster. They can be reached via:

- `dt01.bsc.es`
- `dt02.bsc.es`

These machines share the GPFS filesystem with all other BSC HPC machines. Besides `scp` and `sftp`, they allow some other useful transfer protocols for which in the following usage examples are given:

- `scp`

```
local$ scp <LOCAL_FILE> username@dt01.bsc.es:
local$ scp <USER>@dt01.bsc.es:<REMOTE_FILE> <LOCAL_DIR>
```

- `rsync`

```
local$ rsync -avzP <LOCAL_FILE_OR_DIR> <USER>@dt01.bsc.es:
local$ rsync -avzP <USER>@dt01.bsc.es:<REMOTE_FILE_OR_DIR> \
       <LOCAL_DIR>
```

- `sftp`

```
local$ sftp <USER>@dt01.bsc.es
       username's password:
       sftp> get <REOMTE_FILE>

local$ sftp <USER>@dt01.bsc.es
       username's password:
       sftp> put <LOCAL_FILE>
```

- `BBCP`

```
local$ bbcp -V -z <USER>@dt01.bsc.es:<FILE> <DEST>
local$ bbcp -V <ORIG> <USER>@dt01.bsc.es:<DEST>
```

- `GRIDFTP` (only accessible from dt02.bsc.es)
- `SSHFTP`

```
$ globus-url-copy -help
$ globus-url-copy -tcp-bs 16M -bs 16M -v -vb <YOUR_FILE> \
  sshftp://<USER>@dt01.bsc.es/~/
```

**Data Transfer on the PRACE Network**

Users of the Partnership for Advanced Computing in Europe (PRACE) can utilize the 10Gb/s PRACE Network for moving large data among PRACE sites. To get access to this service, it is required to contact BSC's support (support@bsc.es) requesting its use, providing the machine's local IP from where it will be used.

The selected data transfer tool is `Globus/GridFTP`[4] which is available on `dt02.bsc.es`.

In order to use it, a PRACE user must get access to this machine:

```
$ ssh -l pr1eXXXX dt02.bsc.es
```

Furthermore, it is necessary to load the PRACE environment with:

```
$ module load prace globus
```

[4] Globus/GridFTP http://www.globus.org/toolkit/docs/latest-stable/gridftp/

A proxy certificate needs to be created via:

```
$ grid-proxy-init
  Your identity: /DC=es/DC=irisgrid/O=bsc-cns/CN=john.foo
  Enter GRID pass phrase for this identity:
  Creating proxy ..................................... Done
  Your proxy is valid until: Wed Aug 7 00:37:26 2013
  pr1eXXXX@dtransfer2:~>
```

The command `globus-url-copy` is then available for transferring large data:

```
globus-url-copy [-p <parallelism>] [-tcp-bs <size>] <sourceURL> <destURL>
```

where:

- `-p`: This option specifies the number of parallel data connections to be used (recommended value: 4).
- `-tcp-bs`: This option specifies the size (in bytes) of the buffer to be used by the underlying `ftp` data channels (recommended value: 4MB).
- Common formats for `sourceURL` and `destURL` are:
  - `file://` (on a local machine only)
    (e.g. `file:///home/pr1eXX00/pr1eXXXX/myfile`)
  - `gsiftp://`
    (e.g. `gsiftp://supermuc.lrz.de/home/pr1dXXXX/mydir/`)
  - any URL specifying a directory must end with a frontslash /.

All the available PRACE `GridFTP` endpoints can be retrieved with:

```
$ prace_service -i -f bsc
  gftp.prace.bsc.es:2811
```

More information is available on PRACE's website[5].

### Active Archive management

The Active Archive (AA) is a mid-long term storage filesystem that provides 15 PB of total space. The AA can be accessed from the data transfer machines `dt01.bsc.es` and `dt02.bsc.es` via the mounted GPFS file systems located at `/gpfs/archive/hpc/your_group`.

**Please note:** There is no backup of this file system. The user is responsible for adequately managing the data stored in it.

The special commands `dtcp`, `dtmv`, `dtrsync`, and `dttar` are available to move or copy data from or to the AA. These commands submit a job into a particular partition class, which perform

[5] PRACE GridFTP http://www.prace-ri.eu/Data-Transfer-with-GridFTP-Details

the selected command(s). Their syntax is the same as their corresponding shell commands without the 'dt' prefix (cp, mv, rsync, and tar).

The following commands are available through this method:

- dtq: Shows all the transfer jobs that belong to the user.
- dtcancel: Cancels transfer jobs.
- dttar: Submits a tar-command to queues. The following shows an example on how to tar data from /gpfs to /gpfs/archive/hpc

```
$ dttar -cvf /gpfs/archive/hpc/usertest/outputs.tar ~/OUTPUTS
```

- dtcp: Submits a cp command to queues. To avoid duplicated data, the files in the source tree should be deleted once the copying process to the AA is completed. The following copies data from /gpfs to /gpfs/archive/hpc and vice-versa:

```
$ dtcp -r ~/OUTPUTS /gpfs/archive/hpc/usertest/
$ dtcp -r /gpfs/archive/hpc/usertest/OUTPUTS ~/
```

- dtrsync: Submits an rsync command to queues. To avoid duplicated data, again the files in the source file system should be deleted once the copying process to the AA is completed. The following copies data from /gpfs to /gpfs/archive/hpc and vice-versa:

```
$ dtrsync -avP ~/OUTPUTS /gpfs/archive/hpc/usertest/
$ dtrsync -avP /gpfs/archive/hpc/usertest/OUTPUTS ~/
```

- dtmv: Submits an mv command to queues:

```
$ dtmv ~/OUTPUTS /gpfs/archive/hpc/usertest/
$ dtmv /gpfs/archive/hpc/usertest/OUTPUTS ~/
```

In addition, the above described commands accept the following options:

- --blocking: Block any process from reading the file at the final destination until the transfer is complete.
- --time: Set up a new maximum transfer time (the default is 18h).

These kinds of jobs can be submitted from both the login nodes (automatic file management within a production job) and from the dt01.bsc.es machine. The AA is only mounted on the data transfer machine. Therefore, in case it is desired to navigate through the AA directory tree, navigation has to happen on dt01.bsc.es.

**Repository management (GIT/SVN)**

As explained earlier, outgoing internet connection are not allowed from the cluster. This prevents the use of external repositories directly from the CTS-POWER machines.

As a workaround, the `sshfs` command can be used on the user's local machine. With this command, a desired directory on the GPFS file system can be mounted on the local machine. This allows operating on the mounted GPFS files as if they were stored on the user's local computer. This includes the use of `git`, i.e., cloning, pushing, or pulling any desired repository is possible inside the corresponding mount point. Changes made to any file or directory is directly transferred to the GPFS.

To setup `sshfs`, it is necessary to

- create a directory on the user's local machine that will be used as a mount point,
- and run the following command, where the local directory `<LOCAL_DIR>` is the directory created earlier (note that this command mounts the GPFS home directory by default):

```
$ sshfs -o workaround=rename <USER>@dt01.bsc.es: <LOCAL_DIR>
```

Using this command, the user can directly access the directory. Any modification performed in this directory is automatically replicated to the GPFS file system on the HPC machines. Inside the mounted directory, the user can now call all `git` commands such as `git clone`, `git pull`, or `git push`.

### *2.3.3 Software development on the CTE-POWER system*

All software and numerical libraries available on the cluster can be found in the `/apps/` directory. In case the users need additional software, they can be installed by the support team at BSC (support@bsc.es).

**C/C++ Compilers**

On the cluster, the following `C/C++` compilers are available:

- `xlc/xlc++`: These are the IBM `C/C++` compilers. Help can be obtained from:

```
$ xlc --help
$ xlc++ --help
```

- `gcc/g++`: This is the GNU C/C++ compiler suite for which additional help is available through:

```
$ man gcc
$ man g++
```

- `pgcc/pgc++`: These are the PGI compilers for `C/C++`. To uses these compilers, it is first necessary to load the corresponding module:

```
$ module load pgi
```

Help is available through:

```
$ man gcc
$ man g++
```

All invocations of the `C/C++` compilers follow the suffix conventions for input files listed in Table 2. By default, the preprocessor is run on both C and C++ source files.

| Suffix | Meaning |
|---|---|
| `.C, .cc, .cpp, or .cxx` | `C++` source file |
| `.c` | `C` source file |
| `.i` | preprocessed `C` source file |
| `.so` | shared object file |
| `.o` | object file for the `ld` command |
| `.s` | assembler source file |

**Table 2: Suffix conventions for C/C++.**

The default sizes of the standard `C/C++` datatypes on the CTE-POWER machine are given in Table 3.

| Type | Length (bytes) | Type | Length (bytes) |
|---|---|---|---|
| `bool` (`C++` only) | 1 | `long` | 8 |
| `char` | 1 | `float` | 4 |
| `wchar_t` | 4 | `double` | 8 |
| `short` | 2 | `long double` | 16 |
| `int` | 4 | | |

**Table 3: Default data types available on CTE-POWER.**

The `GCC` version provided by the system is 6.4.0. To better support new and old hardware features, different versions can be loaded via the provided modules. For example, on the CTE-POWER machine, `GCC 9.2.0` can be loaded via:

```
$ module load gcc/9.2.0
```

**Distributed-memory parallelism for C/C++**

To compile Message Passing Interface (MPI) programs, it is recommended to use the handy wrappers `mpicc` and `mpicxx` for `C` and `C++` source code. To uses these wrappers, a parallel environment needs to be chosen first:

```
$ module load openmpi
```

or

```
$ module load ibm_mpi
```

These modules include all the necessary libraries to build MPI applications without having to specify all the details by hand. A compilation of `C` and `C++` programs could look like the following:

```
$ mpicc a.c -o a.exe
$ mpicxx a.C -o a.exe
```

In cases where the user does not want to use the MPI wrappers, the corresponding include files and the libraries can be provided as parameters in the compilation process. To get further information on what needs to be called, the following command can be used for `C`:

```
$ mpicc -show
```

This yields for the `C` PGI compiler:

```
$ pgcc -I/apps/OPENMPI/3.0.0/GCC/include/openmpi \
  -I/apps/OPENMPI/3.0.0/GCC/include -pthread -Wl,-rpath \
  -Wl,/apps/OPENMPI/3.0.0/GCC/lib -Wl,--enable-new-dtags \
  -L/apps/OPENMPI/3.0.0/GCC/lib -lmpi
```

For `C++`, the command is

```
$ mpicxx -show
```

which results in the following output for the PGI compiler:

```
$ pgc++ -I/apps/OPENMPI/3.0.0/GCC/include/openmpi \
  -I/apps/OPENMPI/3.0.0/GCC/include -pthread -Wl,-rpath \
  -Wl,/apps/OPENMPI/3.0.0/GCC/lib -Wl,--enable-new-dtags \
  -L/apps/OPENMPI/3.0.0/GCC/lib -lmpi_cxx -lmpi
```

### Shared-memory parallelism for C/C++

OpenMP directives are supported by the GNU `C` and `C++` compilers. To use shared memory parallelization, the flag `-fopenmp (or -mp)` must be added to the compile line:

```
$ gcc -fopenmp -o exename filename.c
$ g++ -fopenmp -o exename filename.C
```

Furthermore, hybrid parallelism can be realized by mixing MPI and OpenMP code, e.g., by using

```
$ mpicc -fopenmp -o exename filename.c
$ mpicxx -fopenmp -o exename filename.C
```

### FORTRAN compilers

On the CTE-POWER machine the following `FORTRAN` compilers are available:

- `xlf`: This is the IBM `FORTRAN` compiler. Help can be obtained from:

  ```
  $ xlf -qhelp
  ```

- `gfortran`: This is the GNU compiler for `FORTRAN` for which additional help is available through:

  ```
  $ man gfortran
  ```

- `pgfortran`: This is the `FORTRAN` compiler of the PGI for which first a module needs to be loaded:

  ```
  $ module load pgi
  ```

Then, help is available through:

```
$ man pgfortran
```

By default, the compilers expect all FORTRAN source files to have the extension .f, and all FORTRAN source files that require preprocessing to have the extension .F. The same applies to FORTRAN90 source files with the extensions .f90 and .F90.

**Distributed-memory parallelism with FORTRAN**

To use MPI, again wrappers can be used. For FORTRAN, these are the mpif77 and mpif90 wrappers, depending on the source code type. Additional information on the wrappers can be obtained from the manpages, e.g., by calling man mpif77 to see a detailed list of options to configure the wrappers, i.e., to change the default compiler. The wrappers can be used as follows

```
$ mpif77 a.f -o a.exe
$ mpif90 a.f -o a.exe
```

Again, the parameter -show can be specified to get information on what to include and against what to link in case it is desired not to use the MPI wrappers. For example, the output for

```
$ mpif90 -show
```

For the GNU FORTRAN compiler this results in:

```
$ gfortran -I/apps/OPENMPI/4.0.5/GCC/include \
  -pthread -I/apps/OPENMPI/4.0.5/GCC/lib -Wl,-rpath \
  -Wl,/apps/OPENMPI/4.0.5/GCC/lib -Wl,--enable-new-dtags \
  -L/apps/OPENMPI/4.0.5/GCC/lib -lmpi_usempif08 \
  -lmpi_usempi_ignore_tkr -lmpi_mpifh -lmpi
```

OpenMP directives are supported by the GNU/PGI FORTRAN compilers by specifying the additional option -fopenmp (or -mp), e.g.,

```
$ gfortran -fopenmp -o exename filename.f
$ pgfortran -fopenmp -o exename filename.f
```

### *2.3.4 Batch system on the CTE-POWER system*

The CTE-POWER system uses the `SLURM` scheduler[6] for batch processing support. This section provides information for getting started with job execution on the CTE-POWER system.

**Batch jobs**

A job is the execution unit for `SLURM`. Jobs can also be submitted via a job script that holds all the necessary information. Therefore, a job file needs to be created. The job file contains directives describing the job's requirements, and the commands to execute.

To submit jobs with a job file `<job_script>`, the `sbatch` command needs to be used:

```
$ sbatch <job_script>
```

This will return a job id. The status of the job can then be checked with (replace `jobid` by the id of the job):

```
$ squeue --job jobid
```

Jobs can be canceled with:

```
$ scancel jobid
```

In order to ensure the proper scheduling of jobs, execution limitations have been introduced at BSC. There exists a limit in the number of nodes and CPUs that can be used at the same time by a group. These limits can be checked with the `bsc_queues` command, see Sec. 2.3.1. In case an execution that exceeds the limits needs to be run, the user may contact BSC's support (support@bsc.es).

**Interactive Sessions**

An allocation of an interactive session needs to be submitted to the `debug` partition. The following example shows a request for an interactive session with `64` cores and `10` minutes wall clock time:

```
$ salloc -t 00:10:00 -n 1 -c 64 -J debug srun --pty /bin/bash
```

**Job directives**

A job must contain a series of directives to inform the batch system about the job's characteristics. These directives appear as comments in the job script and have to conform to either the `sbatch` syntaxes, i.e., to

[6] SLURM https://slurm.schedmd.com

```
#SBATCH --directive=value
```

Additionally, the job script may contain a set of commands to execute. If not, an external script may be provided with the `executable` directive.

The `debug` partition should be used for testing purposes with small resource allocations:

```
#SBATCH --qos=debug
```

To define the limit of the wall clock time, the following syntax needs to be used:

```
#SBATCH --time=HH:MM:SS
```

This is a mandatory field and it must be set to a value greater than the real execution time for the application and smaller than the time limits granted to the user. It should be noted that the job is killed after the time has passed.

The pathname can be specified by the directive:

```
#SBATCH -D pathname
```

If not specified, it is the current working directory at the time the job was submitted.

To define the name of the files to collect the standard error output (`stderr`) and the standard output (`stdout`) of the job, the following directives are required

```
#SBATCH --error=file
#SBATCH --output=file
```

`SLURM` needs to know the resources to allocate for the job. Therefore, it is necessary to define the number of processes to start. This can be done via the directive:

```
#SBATCH --ntasks=number
```

Optionally, it can be specified how many threads each process would open via:

```
 #SBATCH --cpus-per-task=number
```

The number of CPUs assigned to the job will be the (`total_tasks` number) * (`cpus_per_task` number). To set the number of tasks assigned to a node, the following directive can be used:

```
#SBATCH --ntasks-per-node=number
```

Furthermore, GPUs can be requested as an additional source. The `number` can be specified via the following directive:

```
#SBATCH --gres=gpu:number
```

On the CTE-POWER system, resources on a node can be shared with others. To avoid this, users can request an exclusive use of a compute node by using the directive:

```
#SBATCH --exclusive
```

`SLURM` furthermore offers to use reservations, i.e., to use resources that have previously been granted for executions in a reserved resource set. In a reservation, only a set of accounts can run jobs. This is especially useful for courses. Reservations can be defined in the job script by

```
#SBATCH --reservation=reservation_name
```

To enable email notification, the following set of directives can be used:

```
#SBATCH --mail-type=[begin|end|all|none]
#SBATCH --mail-user=<your_email>
```

Those two directives are presented as a set because they need to be used at the same time. They will enable email notifications that are triggered when a job starts its execution (`begin`), ends its execution (`end`), or both (`all`). The `none` option does not trigger any email, it is the same as not putting the directives. The only requisite is that the email specified is valid and also the same that is used for the HPC User Portal[7]. The following is an example on how to use email notification:

```
#SBATCH --mail-type=end
#SBATCH --mail-user=dannydevito@bsc.es
```

A job can be defined to use `X11` as a graphical interface, i.e., if the user wants to be able to execute a graphical command. If the user does not close the current terminal, a graphical window can be spawned. Using the following directive, `SLURM` assigns the necessary

[7] BSC HPC User Portal https://www.bsc.es/user-support/hpc_portal.php

resources to the job and sets up `X11` forwarding on all, batch host, first or last node(s) of the allocation.

```
#SBATCH --x11=[=<all|batch|first|last>]
```

By default, `SLURM` schedules a job such that a minimum number of switches is used. However, a user can request a specific network topology to run a job. `SLURM` will try to schedule the job for `timeout` minutes. In case `SLURM` is unable to allocate the requested number of switches (from 1 to 14) after `timeout` minutes, `SLURM` will schedule the job with the default options. The directive to specify a network topology is:

```
#SBATCH --switches=number@timeout
```

SLURM environment variables that can be used in the job scripts are shown in Table 4.

| Variable | Description |
|---|---|
| `SLURM_JOBID` | Specifies the job ID of the executing job. |
| `SLURM_NPROCS` | Specifies the total number of processes in the job. |
| `SLURM_NNODES` | This is the actual number of nodes assigned to run the job. |
| `SLURM_PROCID` | Specifies the MPI rank (or relative process ID) for the current process. The range is from 0-(`SLURM_NPROCS`–1). |
| `SLURM_NODEID` | Specifies relative node ID of the current job. The range is from 0-(`SLURM_NNODES`–1). |
| `SLURM_LOCALID` | Specifies the node-local task ID for the process within a job. |

**Table 4: SLURM variables**

### SBATCH examples

An example of a job script for a sequential job could look as follows:

```
#!/bin/bash
#SBATCH --job-name="test_serial"
#SBATCH -D .
#SBATCH --output=serial_%j.out
#SBATCH --error=serial_%j.err
#SBATCH --ntasks=1
#SBATCH --time=00:02:00
./serial_binary> serial.out
```

In contrast, a parallel job requesting more resources with the `--ntasks` and `--cpus-per-task` directives (here 2 GPUs are reserved):

```
#!/bin/bash
#SBATCH --job-name=test_parallel
#SBATCH -D .
#SBATCH --output=mpi_%j.out
#SBATCH --error=mpi_%j.err
#SBATCH --ntasks=16
#SBATCH --cpus-per-task=4
#SBATCH --time=00:02:00
#SBATCH --gres=gpu:2
mpirun ./parallel_binary> parallel.output
```

### SLURM job status and reason codes

When using `squeue`, `SLURM` will report back the status of launched jobs. If the jobs are still waiting to enter execution, a reason will be provided. `SLURM` uses codes to display this information. The most relevant codes are displayed in the following Table 5.

| Code | Meaning |
|---|---|
| `COMPLETED (CD)` | The job has completed the execution. |
| `COMPLETING (CG)` | The job is finishing, but some processes are still active. |
| `FAILED (F)` | The job terminated with a non-zero exit code. |
| `PENDING (PD)` | The job is waiting for resource allocation. The most common state after running `sbatch`, it will run eventually. |
| `PREEMPTED (PR)` | The job was terminated because of preemption by another job. |
| `RUNNING (R)` | The job is allocated and running. |
| `SUSPENDED (S)` | A running job has been stopped with its cores released to other jobs. |
| `STOPPED (ST)` | A running job has been stopped with its cores retained. |

**Table 5: Status codes of SLURM.**

The list in Table 6 contains the most common reason codes of the jobs that have been submitted and are still not in the running state.

| Code | Meaning |
|---|---|
| `Priority` | One or more higher priority jobs is in queue for running. The job will eventually run. |
| `Dependency` | This job is waiting for a dependent job to complete and will run afterwards. |

| Code | Meaning |
|---|---|
| `Resources` | The job is waiting for resources to become available and will eventually run. |
| `InvalidAccount` | The job is waiting for resource allocation. The most common state after running `sbatch`. It will run eventually. |
| `InvaldQoS` | The job's Quality of Service (QoS) is invalid. Cancel the job and resubmit with correct account. |
| `QOSGrpCpuLimit` | All CPUs assigned to your job's specified QoS are in use. The job will run eventually. |
| `QOSGrpMaxJobsLimit` | Maximum number of jobs for your job's QoS have been met. The job will run eventually. |
| `QOSGrpNodeLimit` | All nodes assigned to your job's specified QoS are in use. The job will run eventually. |
| `PartitionCpuLimit` | All CPUs assigned to your job's specified partition are in use. The job will run eventually. |
| `PartitionMaxJobsLimit` | Maximum number of jobs for your job's partition have been met. The job will run eventually. |
| `PartitionNodeLimit` | All nodes assigned to your job's specified partition are in use. The job will run eventually. |
| `AssociationCpuLimit` | All CPUs assigned to your job's specified association are in use. The job will run eventually. |
| `AssociationMaxJobsLimit` | All CPUs assigned to your job's specified association are in use. The job will run eventually. |
| `AssociationNodeLimit` | All nodes assigned to your job's specified association are in use. The job will run eventually. |

**Table 6: SLURM reason codes.**

### Important accounting changes

To ensure fair and reliable CPU usage accounting information, **BSC enforced the need to use at least 40 threads for each GPU requested**. In the job scripts, the number of threads used needs to meet the requirements for the GPU needs. It should be noted that `SLURM` does refer to each thread as if it is a physical CPU.

The value of (`cpu-per-task`) * (`task-per-node`) should amount to those 40 threads. By default, the value of `cpu-per-task` is set to 1.

If the number of tasks in the job cannot be changed, the number of CPUs per task can be edited (`#SBATCH –cpus-per-task=number`). To not affect the executions, the desired CPUs per task can be chosen by setting the environment variable `OMP_NUM_THREADS` (this variable may not work for every application).

Otherwise, an error message will be displayed pointing out this issue:

```
sbatch: error: Minimum cpus requested should be (nodes * gpus/node * 40).
Cpus requested: X. Gpus: Y, Required cpus: Z
sbatch: error: Batch job submission failed: CPU count specification invalid
```

# 3 Nord3 at Barcelona Supercomputing Center

In August 2012, an agreement was signed between the Spanish government and IBM in order to update the supercomputer Marenostrum2[8], and from then on, named Marenostrum3[9]. When Marenostrum4[10] was installed, parts of the machine were moved from the original location and renamed to Nord3. This machine is based on Intel SandyBridge processors, iDataPlex Compute Racks, a Linux Operating System, and an Infiniband interconnection.

In the following, more details of the hardware specifications of the Nord3 system are given in Sec. 3.1. Section 3.2 describes how to access the system before Sec. 3.3 provides a user guideline on how to work with the system.

## 3.1 Hardware specifications

The current peak performance of Nord3 is 251.6 Teraflops per second (TFLOPs). The total number of cores is 12,096 Intel SandyBridge-EP E5–2670 cores, clocked at 2.6 GHz (756 compute nodes) with at least 24.2 TB of main memory.

The following is a summary of the system configuration:

- 9 iDataPlex compute racks, with each one composed of:
  - 84 IBM dx360 M4 compute nodes
  - 4 Mellanox 36-port Managed Fourteen Data Rate (FDR) 10 IB switches
  - 2 BNT RackSwitch G8052F (management network)
  - 2 BNT RackSwitch G8052F (GPFS network)
  - 4 Power distribution units
- All IBM dx360 M4 nodes contain:
  - 2x E5–2670 SandyBridge-EP, clocked at 2.6GHz, with 20MB cache and 8 cores each
  - 500 GB 7200 rpm Serial AT Attachment (SATA) II local Hard Drive Disk (HDD)
  - One of the following Random-Access Memory (RAM) configurations:
    - Default nodes: 32 GB/node
    - Medium memory nodes: 64 GB/node
    - High memory nodes: 128 GB/node
- 15 PB of GPFS disk storage
- Interconnection networks
  - Infiniband Mellanox FDR10: High bandwidth network used by parallel applications communications (MPI)
  - Gigabit ethernet: 10 Gb/s ethernet network used by the GPFS file system.

The operating system is SUSE Linux Enterprise Server 11 SP3. The latest Nord3 system overview can be found online[11].

[8] Marenostrum2 https://www.bsc.es/ca/marenostrum/marenostrum/mn2
[9] Marenostrum3 https://www.bsc.es/marenostrum/marenostrum/mn3
[10] Marenostrum4 https://www.bsc.es/marenostrum/marenostrum
[11] Nord3 system overview https://www.bsc.es/user-support/nord3.php#systemoverview

## 3.2 Accessing the system

The BSC direct collaborators in the project can gain access through the same steps as those described in Sec. 2.2.

Three public login nodes are available to connect to the Nord3 system. The login nodes are:

- `nord1.bsc.es`
- `nord2.bsc.es`
- `nord3.bsc.es`

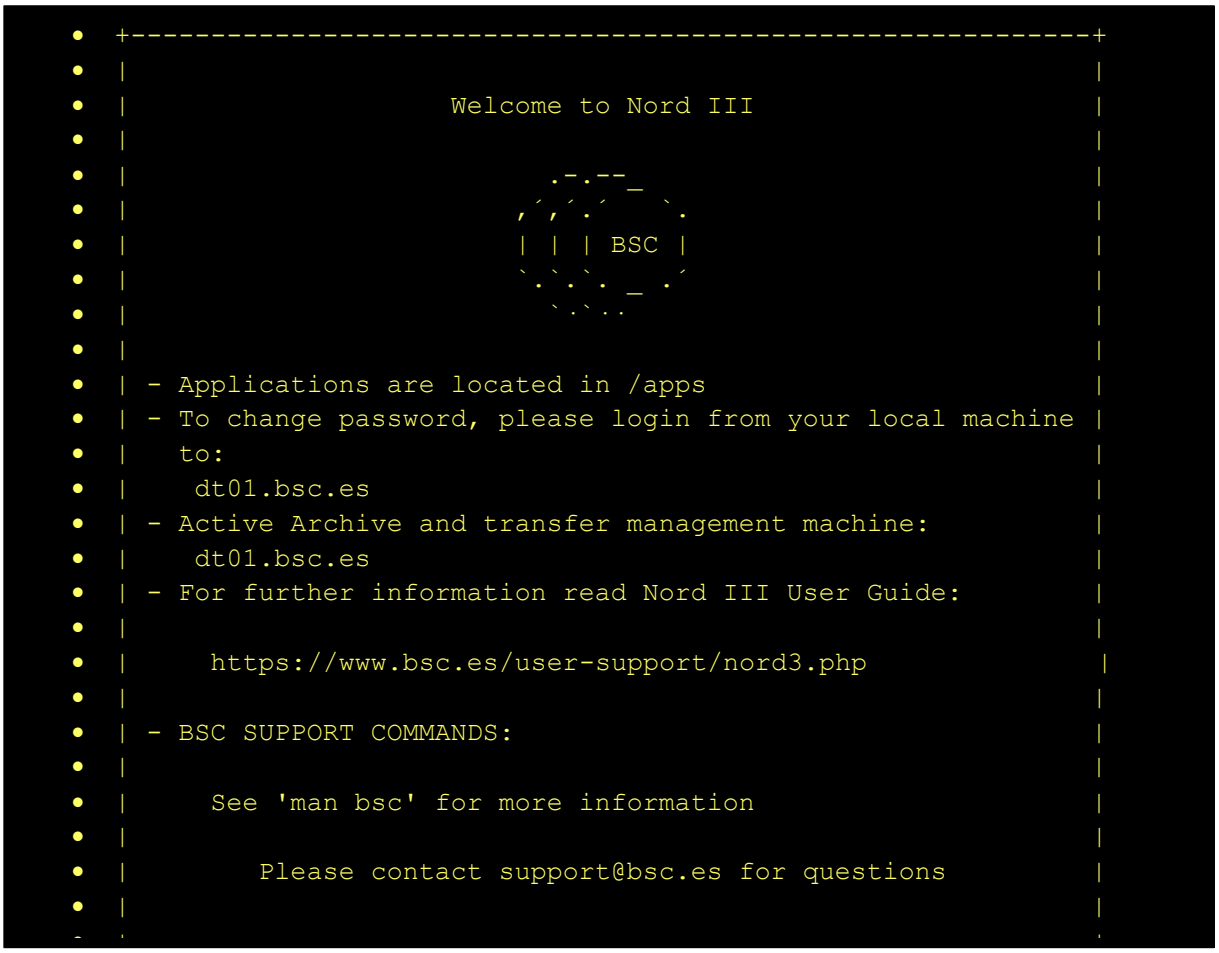



**Figure 3: Nord3 login screen.**

Upon login, the user is provided with a shell on a login node, where code can be compiled, and the applications can be prepared, see Figure 3.

Similar to the CTE-POWER system, see Sec. 2.2.2, the `ssh` tools need to be used to login into the system and to transfer files to the cluster. Incoming connections from protocols like `telnet`, `ftp`, `rlogin`, `rcp`, or `rsh` are not allowed. It should be noted that only incoming connections are allowed on the whole cluster, i.e., once logged in, outgoing connections are rejected for security reasons. Once connected to the machine, the user is presented with a UNIX shell prompt in the home (`$HOME`) directory. Novice users will need to learn the basics before doing anything useful.

It is recommended to change the password on the first login. To change the password, the user has to login to a different machine (`dt01.bsc.es`). The corresponding connection must be established from the user's local machine.

```
local$       ssh -l username dt01.bsc.es
dttransfer1$ passwd
    Changing password for username.
    Old Password:
    New Password:
    Reenter New Password:
    Password changed.
```

The password change takes about 10 minutes to be effective.

## 3.3 Using the system

The Nord3 system uses the same module environment as the CTE-POWER system. For more information on how to work with the module system, the reader is referred to Sec. 2.3.1.

In addition, the following Sec. 3.3.1 gives an overview of the available file systems. Subsequently, Sec. 3.3.2 provides information on how to develop software on Nord3, and Sec. 3.3.3 on how to use the batch system.

### *3.3.1 Available file systems*

**IMPORTANT:** It is in the user's responsibility to backup all critical data. BSC only guarantees a daily backup of user data under `/gpfs/home`. Any other backup should only be done exceptionally if demanded by the interested user.

Each user has several areas of disk space for storing files. These areas may have size or time limits. This section should be read carefully to know about the policy of usage of each of these filesystems.

There are 3 different types of storage available on the cluster:

- The **root filesystem** is the filesystem where the operating system resides. It this is a Network File System (NFS) mounted from one of the servers. It is **NOT** permitted to use the `/tmp` directory for temporary user data. The local hard drive can be used for this purpose (see below).
- The **GPFS** is a distributed networked filesystem, which can be accessed from all the nodes and the data transfer machine. The parallel filesystem GPFS allows parallel applications simultaneous access to a set of files (even a single file) from any node that has the file system mounted while providing a high level of control over all file system operations. An incremental backup will be performed daily only for `/gpfs/home`.
- A **local hard drive** is installed in every node, which can be used as a local scratch space to store temporary files during job execution.

The following Table 7 holds information on the available file systems on Nord3.

| Mount point | Description |
|---|---|
| `/apps` | On this file system, applications and libraries that have already been installed on the machine reside. It is recommended to take a look at the directories to know the applications available for general use. |
| `/gpfs/home` | This file system holds the home directories of all users. Each user has their own home directory to store developed sources and their personal data. A default quota will be enforced on all users to limit the amount of data stored there. **Note:** Jobs should not be run from this file system. Instead, jobs should be executed from the group's `/gpfs/projects` or `/gpfs/scratch` folders. |
| `/gpfs/projects` | For each group of users, there is a directory in `/gpfs/projects`, e.g., the group `bsc01` has the directory `/gpfs/projects/bsc01` ready to use. The space is intended to store data that needs to be shared between the users of the same group or project. A quota per group will be enforced depending on the space assigned by BSC's Access Committee. It is in the project manager's responsibility to determine and coordinate the best use of this space, and to define policies how to distribute or share it between the users. This file system is not mounted on the computing nodes. |
| `/gpfs/scratch` | Each user has a directory on `/gpfs/scratch`. Its intended use is to store temporary files of the jobs during their execution. A quota per group is enforced depending on the space assigned. |
| `/scratch/tmp` | This space is local to the nodes and accessible via the `$TMPDIR` environment variable. The amount of space within the `/scratch` filesystem is about 500 GB. All data stored on the local hard drives of the compute nodes is not available from the login nodes. Local hard drive data are not automatically removed, so each job has to remove its data before finishing. |

**Table 7: Available file systems on Nord3.**

### Quotas

The quotas are the amount of storage available for a user or a groups' users. It can be pictured as a small disk readily available to the user. A default value is applied to all users and groups and cannot be outgrown.

The quota can be inspected at any time by using the following command from inside each filesystem (see also Sec. 2.3.1):

```
$ bsc_quota
```

The command provides a readable output for the quota.

If more disk space on any file system of Nord3 is required, the PI of the project has to make a request for the extra space needed, specifying the requested space and the reasons why it is needed. For more information BSC's support (support@bsc.es) can be contacted.

### *3.3.2 Software development on the Nord3 system*

**C/C++ compilers**

On the Nord3 machine the following `C/C++` compilers are available:

- `icc/icpc`: This is the Intel `C/C++`. Help for these compilers can be obtained from:

```
$ man icc
$ man icpc
```

- `gcc/g++`: These are the GNU compilers for `C/C++` for which additional help is available through:

```
$ man gcc
$ man g++
```

All invocations of the `C` or `C++` compilers follow the suffix conventions as listed in Table 2 in Sec. 2.3.3. By default, the preprocessor is run on both `C` and `C++` source files. The default sizes of the standard `C/C++` datatypes on the machine are the same as on the CTE-POWER architecture. More details can be found in Table 3 in Sec. 2.3.3.

**Distributed-memory parallelism for C/C++**

To compile MPI programs, it is recommended to use the handy wrappers `mpicc` and `mpicxx` for `C` and `C++` source code. To uses these wrappers, a parallel environment needs to be chosen first (either `openmpi`, `impi`, or `poe`):

```
$ module load openmpi
$ module load impi
$ module load poe
```

These modules will include all the necessary libraries to build MPI applications without having to specify all the details by hand. A compilation of `C` and `C++` programs could look like the following:

```
$ mpicc a.c -o a.exe
$ mpicxx a.C -o a.exe
```

In cases where the user does not want to use the MPI wrappers, the corresponding include files and the libraries can be provided as parameters in the compilation process. To get further information on what needs to be called, the following command can be used for `C`:

```
$ mpicc -show
```

This yields, e.g., for the GNU compiler:

```
$ gcc -I/apps/OPENMPI/1.8.1-mellanox/.../hwloc/include \
  -I/apps/OPENMPI/1.8.1-mellanox/.../libevent2021/libevent \
  -I/apps/OPENMPI/1.8.1-mellanox/.../libevent2021/libevent/include \
  -I/apps/OPENMPI/1.8.1-mellanox/include \
  -I/apps/OPENMPI/1.8.1-mellanox/include/openmpi -pthread -Wl,-rpath \
  -Wl,/apps/OPENMPI/1.8.1-mellanox/lib -Wl,--enable-new-dtags \
  -L/apps/OPENMPI/1.8.1-mellanox/lib -lmpi
```

For `C++`, the command is, e.g., for the GNU compiler:

```
$ mpicxx -show
```

which results in the following output:

```
$ g++ -I/apps/OPENMPI/1.8.1-mellanox/.../hwloc/include \
  -I/apps/OPENMPI/1.8.1-mellanox/.../libevent2021/libevent \
  -I/apps/OPENMPI/1.8.1-mellanox/.../libevent2021/libevent/include \
  -I/apps/OPENMPI/1.8.1-mellanox/include
  -I/apps/OPENMPI/1.8.1-mellanox/include/openmpi -pthread -Wl,-rpath \
  -Wl,/apps/OPENMPI/1.8.1-mellanox/lib -Wl,--enable-new-dtags \
  -L/apps/OPENMPI/1.8.1-mellanox/lib -lmpi_cxx -lmpi
```

#### Shared-memory parallelism for C/C++

OpenMP directives are fully supported by the Intel `C` and `C++` compilers. To use shared memory parallelization, the flag `-fopenmp (or -mp)` must be added to the compile line:

```
$ icc -qopenmp -o exename filename.c
$ icpc -qopenmp -o exename filename.C
```

Furthermore, hybrid parallelism can be realized by mixing MPI and OpenMP code, e.g., by using:

```
$ mpicc -fopenmp -o exename filename.c
$ mpicxx -fopenmp -o exename filename.C
```

#### Automatic Parallelization for C/C++

The Intel `C` and `C++` compilers are able to automatically parallelize simple loop constructs, using the option `-parallel`:

```
$ icc -parallel -o a.out a.c
```

#### FORTRAN compilers

On the Nord3 machine, the following `FORTRAN` compilers are available:

- `ifort`: This is an Intel `FORTRAN` compiler. Help can be obtained from:

  ```
  $ man ifort
  ```

- `gfortran`: This is the GNU compiler for `FORTRAN` for which additional help is available through:

  ```
  $ man gfortran
  ```

By default, the compilers expect all `FORTRAN` source files to have the extension `.f`, and all `FORTRAN` source files that require preprocessing to have the extension `.F`. The same applies to `FORTRAN90` source files with the extensions `.f90` and `.F90`.

#### Distributed-memory parallelism with FORTRAN

To use MPI, again wrappers can be used. For FORTRAN, these are the `mpif77` and `mpif90` wrappers, depending on the source code type. Additional information on the wrappers can be obtained from the `manpages`, e.g., by calling `man mpif77` to see a detailed list of options to configure the wrappers, i.e., to change the default compiler. The wrappers can be used as follows:

```
$ mpif77 a.f -o a.exe
$ mpif90 a.f -o a.exe
```

Again, the parameter `-show` can be specified to get information on what to include and against what to link in case it is desired not to use the MPI wrappers. For example, the output for:

```
$ mpif90 -show
```

results in:

```
$ gfortran -D_REENTRANT -I/apps/OPENMPI/1.8.1-mellanox/include \
  -I/apps/OPENMPI/1.8.1-mellanox/lib -Wl,-rpath \
  -Wl,/apps/OPENMPI/1.8.1-mellanox/lib \
  -Wl,--enable-new-dtags -L/apps/OPENMPI/1.8.1-mellanox/lib \
  -lmpi_usempi -lmpi_mpifh -lmpi
```

**Shared-memory parallelism with FORTRAN**

OpenMP directives are supported by the Intel `FORTRAN` compiler by specifying the additional option `-qopenmp`, i.e.,

```
$ ifort -qopenmp a.f -o a.exe
```

**Automatic Parallelization with FORTRAN**

The Intel `FORTRAN` compiler will attempt to automatically parallelize simple loop constructs using the option `-parallel`:

```
$ ifort -parallel a.f -o a.exe
```

### *3.3.3 Batch system on the Nord3 system*

On Nord III, LSF[12] is used for batch processing support, i.e., all jobs must be run through it. The subsequent documentation provides information for getting started with job execution on Nord3.

**LSF commands**

To submit a job to LSF, the following command can be used:

```
$ bsub < job_script
```

where `job_script` is a file containing the job directives[13]. The file is passed through standard input (`stdin`) to the queue system.

[12] BSC LSF documentation http://www.bsc.es/support/LSF/9.1.2
[13] LSF job directives https://www.bsc.es/user-support/nord3.php#jobdirectives

The `bjobs` command allows to show all submitted jobs:

```
$ bjobs [-w][-X][-l job_id]
```

To remove a job from the queueing system or to cancel the processes' execution, if they are still running, the following command can be used (`<job_id>` is the id of the job, which can be obtained from the `bjobs` command):

```
$ bkill <job_id>
```

There are specific BSC commands available (see further below and on BSC's website[14]). To show all the pending or running jobs from the user's group, the following command can used:

```
$ bsc_jobs
```

### Interactive Sessions

The allocation of an interactive session on the interactive partition has to be done by executing

```
$ bsub -q interactive -W 01:00 -n 1 -Is /bin/bash
```

**NOTE:** that an interactive session's run limit is preset to 4 hours (`-W 04:00`).

### Job directives

A job must contain a series of directives to inform the batch system about the job's characteristics. The user is encouraged to read the `bsub` command's manual on any of Nord3's terminals:

```
$ man bsub
```

In the following, a short summary of most common directives is provided.

To specify the name (description) of the job, the following directive needs to be used:

```
#BSUB -J job_name
```

The queue for the job to be submitted to can be specified with:

```
#BSUB -q debug
```

[14] Nord3 BSC commands https://www.bsc.es/user-support/nord3.php#bsccommands

The `debug` queue is only intended for small tests, i.e., there is a limit of 1 job per user, using up to 64 CPUs (4 nodes), and one hour of wall clock limit. The queue might be reassigned by LSF's internal policy, as with the sequential queue[15].

To specify how much time the job will be allowed to run, the following mandatory directive needs to be specified:

```
#BSUB -W HH:MM
```

**NOTE:** It is not possible to specify the number of seconds in LSF. The time must be set to a value greater than the real execution time for the application and smaller than the time limits granted to the user. The job will be killed after the elapsed period.

To set the working directory of the job, i.e., where the job will run, the subsequent directive needs to be specified (if not specified, it is the current working directory at the time the job is submitted):

```
#BSUB -cwd pathname
```

To set the names of the files to collect the `stderr` and `stdout` outputs of the job the directives:

```
#BSUB -e/-eo file
#BSUB -o/-oo file
```

should be used. For the error file, `%J` as a place holder for job_id can be used. Here, the `-e` option will append the output to the file, while the `-eo` option will replace the file. In contrast, for the output file, the `-o` option will append the output to the file, while the `-oo` will replace the file.

To set the number of tasks for the job, the `-n` option needs to be used:

```
#BSUB -n number
```

For MPI executions, this corresponds to the number of MPI processes. For sequential executions, the option defines the number of cores.

By default, nodes are used exclusively, except for sequential executions. To force LSF to exclusively reserve a node, the following directive needs to be used:

[15] LSF internal policy for sqeuntial jobs https://www.bsc.es/user-support/nord3.php#sequentialexecutions

```
#BSUB -x
```

To specify the minimum amount of memory in MB necessary for each task, the `-M` option should be used:

```
#BSUB -M number
```

This option is used to schedule the jobs better and to select the compute nodes adequate to fulfil the job's needs. By default, 1,800 MB are reserved per task. Exclusive jobs will still get all available memory.

Leaving this option on its defaults, the algorithm could choose any kind of node. The minimum amount of necessary memory per task (in MB) can be requested with the `-M` and the number of tasks with the `-n` options. If the multiplication of these values is greater than 32 GB, then nodes with 64 GB or 128 GB of memory will be reserved. In case a value greater than 64 GB is specified, nodes with 128 GB of memory will be allocated.

Here is a list of the common parameters needed to request each type of node:

- 32 GB/node (default: `#BSUB -M 1800`)
- 64 GB/node (medium memory: `#BSUB -M 3000`)
- 128 GB/node (high memory: `#BSUB -M 7000`)

The user may check the FAQ[16] for a more in-depth explanation. **NOTE:** For non-LowMem requests a ptile of 16 must be specified.

To set a certain reservation with id `reservation_ID`, where the jobs will be allocated, the following directive can be used:

```
#BSUB -U reservation_ID
```

This assumes that the user has access to such a reservation. In some occasions, node reservations can be granted for executions, where only a set of accounts can run jobs. This is especially useful for courses.

The next directive sets the number of processes assigned to a node:

```
#BSUB -R "span[ptile=number]"
```

It should be noted that full nodes will be allocated except for sequential executions. The following is a job script example using this directive:

[16] BSC Nord3 FAQ http://www.bsc.es/user-support/faq.php#therearesomespecialrequirementsforjobs.howshouldiputthem

```
# if 4 processes per node and 4 threads should be used:
#BSUB -R "span[ptile=4]"
export OMP_NUM_THREADS=4

# if the program has high memory consumption
# the number of processes per node can be reduced
#BSUB -R "span[ptile=14]"
```

**MPI particulars**

There are different MPI implementations available for usage, see Sec. 3.3.2. Some of them require special job directives and execution commands. Subsequently, job script examples are presented for the OpenMPI, Intel MPI and IBM POE MPI implementations:

- OpenMPI:

```
#!/bin/bash
#BSUB -n 128
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err
#BSUB -J openmpi_example
#BSUB -W 00:05

module load openmpi
mpirun binary.exe
```

- IntelMPI:

```
#!/bin/bash
#BSUB -n 128
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err
#BSUB -J impi_example
#BSUB -W 00:05

module load impi
mpirun binary.exe
```

- IBM POE:

```
#!/bin/bash
#BSUB -n 128
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err
#BSUB -J poe_example
#BSUB -W 00:05
#BSUB -a poe

module load poe
poe binary.exe
```

**Jobscript examples**

In the following, further job script examples are given.

An example of a job script submitting a sequential job could look as follows:

```
#!/bin/bash
#BSUB -n 1
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err
#BSUB -J sequential
#BSUB -W 00:05

./serial.exe
```

In case OpenMP should be used for shared-memory parallelization, the `OMP_NUM_THREADS` option and the `-n` directive need to be specified:

```
#!/bin/bash
#BSUB -n 16
#BSUB -R "span[ptile=16]"
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err
#BSUB -J sequential_OpenMP
#BSUB -x
#BSUB -W 00:05

export OMP_NUM_THREADS=16
./serial.exe
```

For a parallel execution using MPI, a job file could look as follows:

```
#!/bin/bash
#BSUB -n 128
#BSUB -o output_%J.out
#BSUB -e output_%J.err

# In order to launch 128 processes with 16 processes per node:
#BSUB -R "span[ptile=16]"
#BSUB -J WRF.128-4
#BSUB -W 02:00

# The parallel environment is chosen through modules
module load intel openmpi

mpirun ./wrf.exe
```

To combine MPI and OpenMP threads, the following example can be useful:

```
#!/bin/bash
# The total number of processes:
# 128 MPI processes + 2 OpenMP threads/process = 256 cores
#BSUB -n 128              # processes
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err

########################################################
# This will allocate 8 processes per node, i.e.,     #
# there will be 8 cores per node for the threads     #
########################################################
#BSUB -R "span[ptile=8]"

# exclusive mode (enabled by default)
#BSUB -x

########################################################
# Then (128 MPI tasks) / (8 tasks/node) = 16 nodes  #
# reserved                                           #
# 16 nodes * 16 cores/node = 256 cores allocated    #
# (Matches the amount of cores asked for above)     #
########################################################
#BSUB -J WRF.128-4
#BSUB -W 02:00

# Clean the environment modules
module purge

# The parallel environment is chosen through modules
module load intel openmpi

# 8 MPI processes per node and 16 cpus available
# (2 threads per MPI process):
export OMP_NUM_THREADS=2

mpirun ./wrf.exe
```

The following example shows how to ask for 3,000 MB/task instead of the default 1,800 MB/task. As no `-R` options are included, it will still put 16 processes on each node, i.e., the Common (2 GB/core) nodes will not be able to execute this job. That effectively guarantees that either MedMem (4 GB/core) or HighMem (8 GB/core) nodes will be used, or a mixture of both.

```
#!/bin/bash
# Total number of tasks
#BSUB -n 128
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err

# Requesting 3000 MB per task
# As no ptile is specified, only Medium and High memory nodes
# are elegible for this execution (max 128 nodes, 2048 cores)
#BSUB -M 3000

#BSUB -W 01:00

module purge
module load intel openmpi

mpirun ./my_mpi.exe
```

The following example requests the same memory per task as the preceding one, but it is set to be executed on any nodes. This is done by reducing the number of tasks per node such that the total memory requested by all the tasks in the same node is below the LowMem memory threshold. This is necessary for jobs that need more nodes or CPUs than the MedMem and HighMem nodes can provide.

```
#!/bin/bash

# Total number of tasks
#BSUB -n 128
#BSUB -oo output_%J.out
#BSUB -eo output_%J.err

# Requesting 3000 MB per task
#BSUB -M 3000

# Only 9 task per node.
# All compute nodes are available for execution
#BSUB -R "span[ptile=9]"

#BSUB -W 01:00

module purge
module load intel openmpi

mpirun ./my_mpi.exe
```

### Queues

There are several queues present in the machines and different users may access different queues. All queues have different limits in the number of cores for the jobs and duration. The user can check all accessible queues and their limits anytime using:

```
$ bsc_queues
```

### BSC Commands

The support team at BSC provides some commands useful for the user's awareness and ease of use of the HPC machines. These commands are available through a special module (`bsc/current`) loaded at the beginning of any session. A short summary of these commands follows:

- `bsc_acct`: Displays accounting information on the project's allocation usage.
- `bsc_jobcheck`: Returns a comprehensive description of the resources requested by a jobscript.
- `bsc_jobs`: Shows a list of your submitted jobs and those from other members of the users's group.
- `bsc_load`: Shows the performance and resources usage of all the nodes of a specific running job.
- `bsc_queues`: Shows the queues the user has access to and their time/resources limits.
- `bsc_quota`: Shows a comprehensible quota usage summary for all accessible filesystems.

All available commands have a dedicated `manpage` (not all commands are available for all machines). More information on these commands can be obtained by checking their respective `manpage`:

```
$ man <bsc_command>
```

### Sequential executions

For any job that requires a node or fewer resources, the sequential queue is automatically applied. This queue is the only one that uses the same node for more than one job at a time. It also has the least priority on the machine, and the number of concurrently executed sequential jobs is limited to avoid disturbing large job executions.

If 16 processes per node are requested or the `-x` option is used, the full node will be assigned to the job. If there appear memory problems, using the exclusive flag may solve the problem. Just setting a ptile may still share the node with other users.

# 4 JUWELS and JURECA at Forschungzentrum Jülich

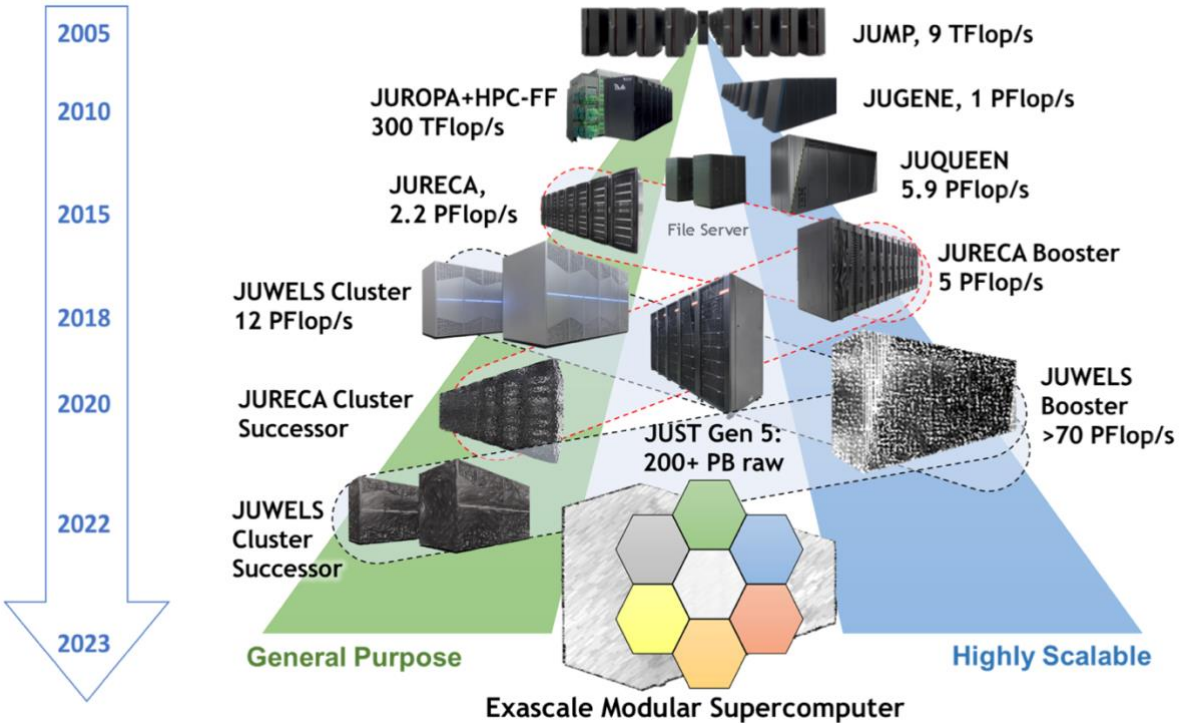



**Figure 4: Past and present supercomputing landscape at the Jülich Supercomputing Centre and roadmap towards exascale computing.**

The continuously increasing complexity of computations with many different concurrently executed functionalities requires to strive for a strategy to use hardware under the constraints of minimal energy consumption and minimal time to solution. Since 2005, JSC at FZJ follows the strategy of providing a mixture of general-purpose architectures (GPAs) and highly-scalable systems to HPC users, see Figure 4. It provides flexibly usable CPU-based GPAs and Booster architectures that allow applications to scale to full system size. It thereby fulfils the continuously growing demands of the broad spectrum of applications that require HPC from science and industry. At present (2021), the two systems JUWELS and JURECA unite the GPA- and Booster-approaches on a production level. The fundamentals for this kind of HPC machines, i.e., the Modular Supercomputing Architecture (MSA), was initially developed in the DEEP project series[17]. The MSA enables to efficiently support computations with heterogenous functionalities by enabling to place specific workflow components on the best possible hardware to obey the constraints of minimal energy consumption and minimal time to solution. Figure 5 shows the general setup of an MSA with Cluster Nodes (CN) and Booster Nodes (BN) connected via a network. The MSA can be used by applications with diverse hardware requirements.

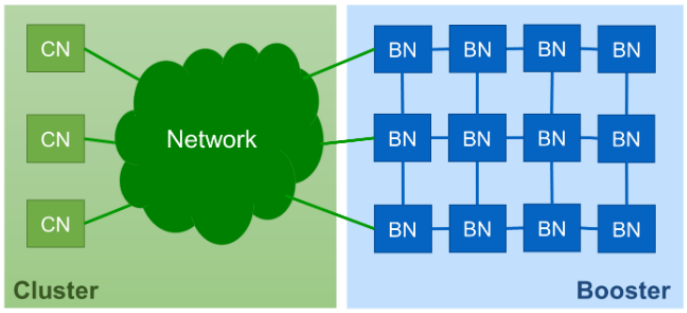



**Figure 5: Cluster-Booster setup in the Modular Supercomputing Architecture (MSA) environments of JUWELS and JURECA at JSC.**

[17] DEEP projects https://www.deep-projects.eu

In the following, the hardware specifications of the JUWELS and JURECA systems are provided in Sec. 4.1. Subsequently, Sec. 4.2 discusses how to access the system, before Sec. 4.3 explains how to use the systems efficiently. For continuously updated documentation, the reader is referred to the JUWELS[18] and JURECA[19] documentations.

## 4.1 Hardware specifications

For the CoE RAISE, the two production systems JUWELS and JURECA are relevant for developments. Their hardware specifications are described in the following Sec. 4.1.1 and Sec. 4.1.2.

### *4.1.1 JUWELS*

The JUWELS system consists of the following two components:

- The **JUWELS Cluster Module** consists of in total of 2,583 nodes. 2,511 nodes are equipped with two Intel Xeon Platinum 8168 CPUs (24 cores per CPU), clocked at 2.7GHz. 2,271 of these nodes feature 96 GB of DDR4 RAM, while 240 fat nodes feature 192 GB of Double Data Rate (DDR) 4 RAM. Further 56 nodes are equipped with two Intel Xeon Gold 6148 CPUs (20 cores per CPU), clocked at 2.4 GHz, and contain each four NVIDIA V100 GPUs with 16 GB HBM and 192 GB of DDR4 RAM. All the aforementioned nodes are operated diskless and are interconnected via an InfiniBand EDR (Connect-X4) connection. In addition, there exist 12 login nodes, each with two Intel Xeon Gold 6148 CPUs, 768 GB of DDR4 RAM, and two 1 TB HDDs configured as Redundant Array of Independent Disk (RAID) 1. The login nodes are interconnected via an InfiniBand EDR (Connect-X5) and a 100 GB/s ethernet network. For visualization work, further four nodes with a similar configuration as the login nodes exist. Unlike the login nodes, each of them carries an NVIDIA Pascal P100 GPU.

  In total, 122,768 CPU cores are available with a computing power of 10.6 (CPU) + 1.7 (GPU) PFLOPs peak performance. The network is a Mellanox InfiniBand EDR fat-tree network with 2:1 pruning at leaf level and top-level HDR switches, which allows a 40 Tb/s transfer rate to the JUWELS Booster Module (see below) and a 250 GB/s network connection to the file system of the Jülich Storage Cluster (JUST) for storage access.

- The **JUWELS Booster Module** with in total 940 nodes, each equipped with two AMD EPYC Rome 7402 CPU (24 cores per CPU), clocked at 2.7GHz. Four of the nodes are login nodes with 768 GB DDR4 RAM and interconnected via an EDR-Infiniband (Connect-X4). The remaining 936 nodes are the compute nodes, which are each equipped with 512 GB DDR4 RAM and four NVIDIA A100 GPUs with four 40 GB HBM2. Each compute node has four InfiniBand HDR (Connect-X6) connections. All Booster Module nodes are operated diskless.

  In total, the Booster Module, with its 3,744 GPUs delivers a peak performance of 73 PFLOPs. The nodes are interconnected via a Mellanox InfiniBand HDR network with a DragonFly+ topology with 20 cells. They allow a 40 Tb/s connection to the JUWELS Cluster Module and a 350 GB/s connection to the JUST system for storage access.

[18] JUWELS documentation https://apps.fz-juelich.de/jsc/hps/juwels/index.html
[19] JURECA documentation https://apps.fz-juelich.de/jsc/hps/jureca/index.html

### *4.1.2 JURECA*

The JURECA system consists of the following two components:

- The **JURECA DC Cluster Module** consists of in total 780 nodes, all equipped with two AMD EPYC 7742 CPUs (64 cores per CPU), clocked at 2.25 GHz. 480 of the 780 compute nodes feature 512 GB of DDR4 RAM, while 96 fat nodes have 1,024GB of DDR4 RAM. Further 192 accelerated compute nodes have 512 GB of DDR4 RAM and four NVIDIA A100 GPUs with four 40 GB HBM2e. The remaining 12 nodes are login nodes, each equipped with 1,024 GB DDR4 RAM and two NVIDIA Quadro RTX8000 GPUs. The accelerated nodes are connected via two InfiniBand HDR (Connect-X6) connects to the remaining system while all other nodes are connected via a single connect of the same type. The login nodes have an additional 100 GB/s ethernet external connection.

  In total, with its 98,304 CPU cores and 768 GPUs, the system has 3.54 (CPU) + 14.98 (GPU) PFLOPs peak performance. The network topology is a Mellanox InfiniBand HDR (HDR100/HDR) DragonFly+, which connects with roughly 15 Tb/s to the JURECA Booster Module (see below) via gateway nodes and with 350 GB/s to the file system on JUST for storage access.

- The **JURECA Booster Module** consists of 1,640 compute nodes, each equipped with an Intel Xeon Phi 7250-F Knights Landing (KNL) accelerator with 68 cores, clocked at 1.4GHz. Each node has 96 GB of memory plus 16 GB Multichannel DRAM (MCDRAM) high-bandwidth memory.

  The Booster Module has with its 111,520 KNL cores a peak performance of 5 PFLOPs. The nodes are connected via an Intel Omni-Path Architecture high-speed network with non-blocking fat-tree topology capable of a >100 GB/s storage connection to JUST.

## 4.2 Accessing the systems

To use computing resources at JSC, it is necessary to have a compute-time project. New users can register to gain access to such a project and to use the allocated resources. In the following, first an overview on how to apply for resources at JSC is given in Sec. 4.2.1, before details on the creation of a user account and the assignment to a project are given in Sec. 4.2.2 and Sec. 4.2.3. Finally, Sec. 4.2.4 explains how to login to the systems. The descriptions are similar to those found in the “Best practice guidelines / tutorials prototype” document of the CoE RAISE. However, the prototypes are directly accessible to the developers in RAISE and do not require to file a compute-time application.

### *4.2.1 Compute-time applications*

Applications for resources on JUWELS and JURECA have to be filed by scientists. Eligible are those who are employed at universities or research facilities in Germany; here, the nationality of the applicant, i.e., the PI, is irrelevant. Applicants from Europe but outside of Germany can apply at PRACE[20], see below. Individual cases may deviate from this general rule after consultation with the Scientific Council and the Board of Directors of the John von Neumann-Institute for Computing (NIC)[21].

[20] PRACE https://prace-ri.eu
[21] NIC http://www.john-von-neumann-institut.de

The PI must have a proven scientific record (preferable a Ph.D. or comparable degree) and must be able to accomplish the proposed tasks. Computing resources are allocated on the basis of reports of independent reviewers. Apart from the project's scientific relevance, the efficient use of the requested supercomputer by the project is an important criterion for the allocation of computing time. That is particularly relevant to the use of a large number of processors in parallel. Computing time periods are yearly - with applications possible twice a year - and begin 1 May and 1 November.

Further information on how to apply for computing time can be found on JSC's website[22].

**Computing Time: Core-hours and EFLOP**

Applicants may request computing time for several systems at the same time within a single proposal. Supercomputers operated at JSC consist of modules which differ considerably in their architectures. So far, computing time has been requested exclusively in units of core-hours. However, the effective computing power of the modules per core-hour differs substantially due to their individual architectures. Thus, computing time requirements for more than one module cannot be specified coherently by adding up core-hours over the requested modules. To take this into account, resource requirements for each resource type, i.e., each specific machine, in core-hours will be converted to floating-point operations in units of Exaflops (EFLOP) based on the theoretical peak performance of the requested module. This reflects the amount of $10^{18}$ floating-point operations per year available to approved projects. The project's total requirement is given by adding up the requested computing time in EFLOP for all modules. In the electronic questionnaire, computing time requirements have to be specified per resource type either, as previously, in million core-hours (Mcore-h) or in EFLOP. Both units will be converted into each other on the fly and displayed in due consideration of the requested resource types. Nonetheless, resources will be granted, allocated and accounted in core-hours only for each resource type. Further information can be taken from the fact sheets below.

**Applications for GCS Large-Scale Projects and GCS/NIC Regular Projects on JUWELS and/or JURECA Booster Module**

The Gauss Centre for Supercomputing (GCS)[23] computing time call is open to all applicants who are employed at universities or research facilities in Germany. Eligible applicants may apply for computing time on JUWELS and/or JURECA Booster Module.

**Please note:** Scientists from FZJ and RWTH are explicitly invited to submit proposals for GCS/NIC Regular as well as GCS Large-Scale projects. However, computing time on the JURECA Booster is only available via the Jülich Aachen Research Alliance (JARA)[24] for those applicants as described below.

Large-Scale projects are projects that require a large number of computing resources over longer periods of time. In detail, projects are classified as "Large-Scale" if they require at least 2% of the systems' annual production in terms of estimated availability. Projects which do not fall into the category "Large-Scale" are called GCS Regular projects.

[22] How to apply for computing time https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/ComputingTime/computingTime_node.html
[23] GCS https://gauss-centre.eu
[24] JARA https://www.jara.org/en/

**Applications for Computing Time on JURECA via JARA**

Scientists who are affiliated with FZJ or RWTH Aachen University are eligible to apply for computing time on JURECA. Additionally, resources on the CLAIX system at RWTH Aachen University are offered in the frame of the JARA Computing Time Call.

**Please note:** The JARA Call is the only possibility for FZJ and RWTH Aachen University researchers to apply for resources on the JURECA Booster module. Furthermore, resources on the JURECA-DC module are exclusively offered for scientists of FZJ.

**Applications through PRACE**

On a European level, applications are also accepted through PRACE. More information on the calls for proposals can be found on the PRACE website (see above).

**Application for Test or Preparatory Access[25]**

Computational science today is tackling problems of increasing complexity and scale. To fulfil the requirements of computationally intensive simulations and analyses of large data sets, applications codes suitable for HPC systems need to be newly developed or adapted from algorithms initially designed for PCs and small clusters. Besides scalability issues from distributed memory, users are also confronted with adapting their applications to various hybrid programming models, including accelerators such as GPU and Intel Many-Integrated Core (MIC) architectures.

To further strengthen the ongoing efforts to meet this challenge and to encourage computational scientists to port their models to state-of-the-art HPC machines to open up new perspectives for their research, JSC invites potential new users of the supercomputers in Jülich to apply for test accounts or preparatory access to the JURECA cluster and booster systems, or JUWELS.

Analogously to similar schemes previously introduced within PRACE, the JSC Preparatory Access aims to facilitate access to the Jülich supercomputers for researchers with computationally intensive scientific problems, but codes that still need to be improved for HPC readiness prior to a full NIC/GCS or JARA-HPC/VSR proposal. Successful applications will receive a computing time budget on the HPC resource, optionally supplemented by expert assistance from one of the JSC Simulation and Data Laboratories (SDLs)[26] for a period of up to four months. The goal is to verify and improve the performance of their application codes and prepare a full computing-time proposal by the next call deadline (February or August).

### *4.2.2 User-account creation*

Gaining access to computational resources at JSC has been standardized in 2018 for all available systems. New users can register anytime through the web portal provided by JuDoor[27]. The system allows managing personal user data and accounts on the HPC systems at JSC and the corresponding project memberships.

[25] Preparatory Access https://www.fz-juelich.de/ias/jsc/EN/Expertise/SimLab/PrepAccess/PrepAccessNode.html
[26] SDLs https://www.fz-juelich.de/ias/jsc/EN/AboutUs/Organisation/ComputationalScience/_node.html
[27] JuDoor https://judoor.fz-juelich.de

To create a new account on JSC's systems, it is necessary to register on JuDoor's login website, see Figure 6 – step 1. By clicking the "Register" link, the user is transferred to a form, where an email address needs to be provided (Figure 6 – step 2). An email containing a registration link is then sent to the user. The user is transferred to a registration form by following this link, where additional personal data needs to be provided. After completion, an account is created, and the user can from now on login to the JuDoor system.

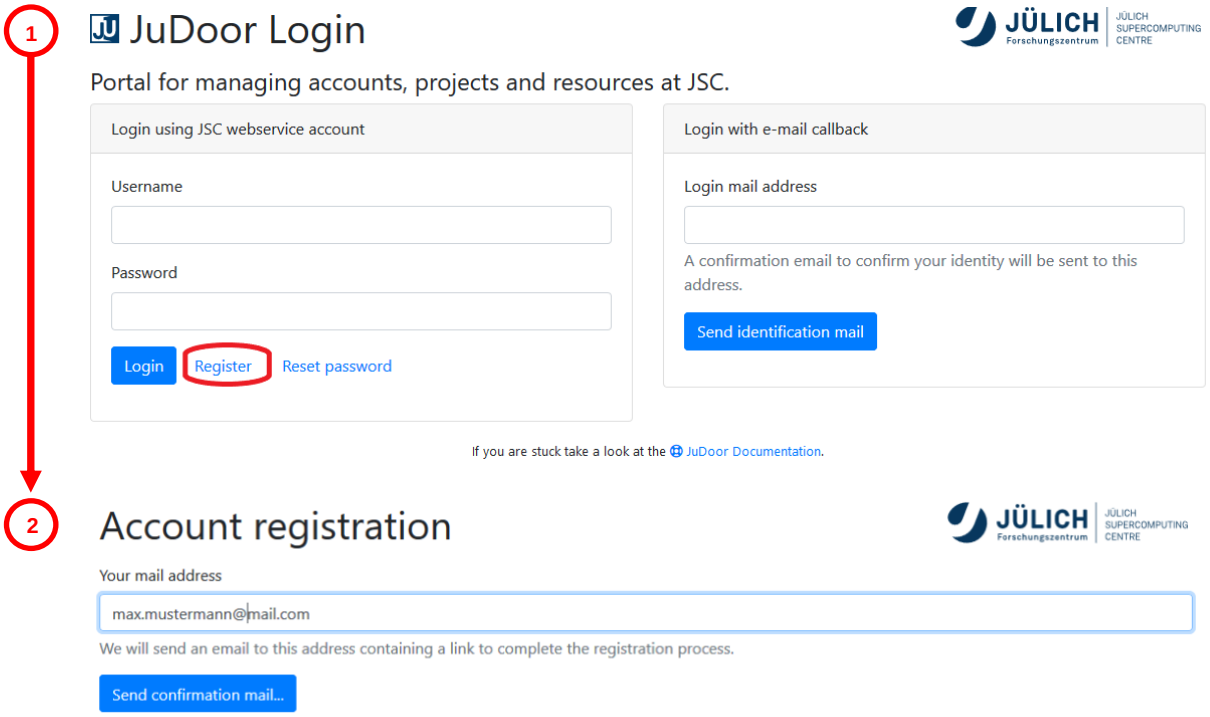



**Figure 7: User-account registration using JuDoor.**

To be assigned to a compute-time project, the user needs to join a project in JuDoor. In the section "Projects" of JuDoor's main view, the "Join a project" button needs to be clicked, see Figure 7 – step 3. This opens the form shown in Figure 7 – step 4. Here, a project id needs to be provided, which can be obtained from the PI having a running compute-time project at JSC. The identifier "TypeX_Y" in Figure 7 is a placeholder for this project id. Additionally, further information can be provided. By clicking the "Join project" button, the user's request is sent to the corresponding project's PI/PA. The user then has to wait until the PI/PA grants access to resources on the system(s).

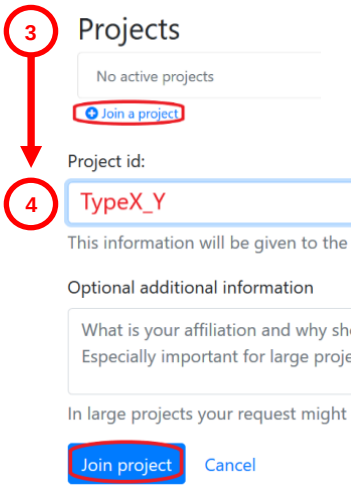



**Figure 6: Joining a project in JuDoor.**

After resources have been granted, new HPC systems will appear in the section “Systems”. For each of them it is necessary to sign the usage agreement, see Figure 8.


**Figure 8: Signing the usage agreement for accessible systems.**

### *4.2.3 Project assignment in JuDoor*

The PI of a compute-time project that has been granted computational resources on JSC’s HPC machines can manage the project members online. The PI also can assign a project administrator (PA) who can manage accounts on the PI’s behalf.

Upon a join request from a user, the PI and the PA receive an email, asking them to confirm the user assignment. The email contains a link that leads, after authenticating against JuDoor, to a website. The PI/PA can confirm the project members and assign available resources to the user.

### *4.2.4 Logging into the system*

Both the JUWELS and JURECA system are available via `ssh`. Before `ssh` can be used, a public `ssh` key needs to be uploaded. The procedure and the security constraints, together with how to login to JSC’s systems, are described subsequently.

After the user agreement, see Sec. 4.2.2, has been signed, a new link “Manage SSH-keys” right next to the project appears in JuDoor’s main view. The corresponding website already provides some examples on how to fill the form, see Figure 9. Due to the European-wide security incident in 2020, the security policies for `ssh` changed at JSC. Private `ssh` keys are not allowed anymore on JSC’s systems and the public key information needs to carry a `from`-clause.

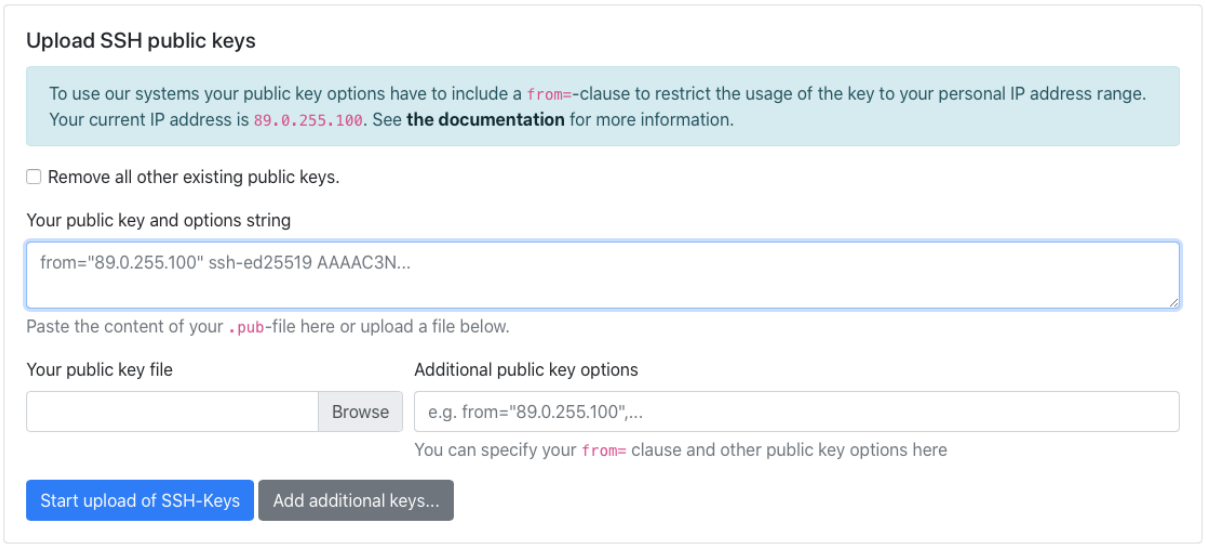



**Figure 9: Upload form for `ssh` keys in JuDoor.**

To generate a `ssh` key on a Linux or MacOS system, open a terminal application and type:

```
$ ssh-keygen -a 100 -t ed25519 -f ~/.ssh/id_ed25519
```

This will create a private / public key pair `id_ed25519` / `id_ed25519.pub` in the folder `~/.ssh/` with key type `ed25519`. In the generation process, the user is asked to provide a password to protect the key. Here it is necessary to ***not*** leave the password empty. The content of the public key file, i.e., `~/.ssh/id_ed25519.pub` then needs to be copied to "Your public key and options string" field in the "Manage SSH-keys". Alternatively, the key needs to be uploaded via the "Your public key file" field. In case a Windows-based OS is used, we recommend reading the `ssh` tutorial on www.ssh.com[28], which explains how to use `Putty` as a `ssh` client and for a key pair generation.

The content of the `from`-clause, the `89.0.255.100` in the example in Figure 9, is a comma-separated list of IPs or hostnames, which can also include partial patterns. If one of the patterns matches, access to the system is granted. The following options for patterns are possible:

- A literal IP address, like `134.94.7.247` (**Note**: the HPC systems can only be reached via IPv4)
- A literal hostname, like `host.example.com`
- An IP or hostname with wildcard operators, like `*.example.com`
- An IP range in Classless Inter-Domain Routing (CIDR) notation, like `134.94.0.0/16`

For more information on how to generate keys and to specify key option strings, the reader is referred to the JUWELS access documentation[29].

Once the public key has been uploaded, the systems can be accessed via `ssh` after a few minutes (replace `user` by your username):

```
$ ssh -i ~/.ssh/id_ed25519 user@juwels.fz-juelich.de
```

or

```
$ ssh -i ~/.ssh/id_ed25519 user@jureca.fz-juelich.de
```

For easier access, it is recommended to write a `config` file in your `~/.ssh/` folder with the following content (replace again `user` by your username):

```
Host juwels
      HostName juwels.fz-juelich.de
      User user
      IdentityFile ~/.ssh/id_ed25519
```

[28] SSH Putty tutorial https://www.ssh.com/ssh/putty/windows/puttygen
[29] JUWELS access documentation https://apps.fz-juelich.de/jsc/hps/juwels/access.html

or by:

```
Host jureca
      HostName jureca.fz-juelich.de
      User user
      IdentityFile ~/.ssh/id_ed25519
```

Then, the login command can be exchanged by:

```
$ ssh juwels
```

or by:

```
$ ssh jureca
```

To avoid entering the key password every time the key is used, the key can furthermore be added to the keychain by the following command:

```
$ ssh-add ~/.ssh/id_ed25519
```

## 4.3 Using the systems

This section describes how the two systems JUWELS and JURECA can be used. Some of the mechanisms are identical on both machines, e.g., the software stack usage or the available file systems, while others such as the scheduler's partition names are different.

First, Sec. 4.3.1 gives an introduction on how to use the software stacks and the module systems. Subsequently, Sec. 4.3.2 provides an overview of the existing file systems and explains how to transfer files from and to the HPC machines. Finally, Sec. 4.3.3 and Sec. 4.3.4 deliver information on how to develop software on the systems and how to use the batch systems.

### *4.3.1 Software stacks and introduction to the module systems*

Both the JUWELS and JURECA systems use the `Easybuild`[30] module system to change programming environments and libraries easily. The modules are organized hierarchically, and on login a standard set of modules is loaded. To show the loaded modules, the user can use the command

```
$ module list
```

To list all available modules the command

```
$ module avail
```

can be executed. This command's output is usually split into different module categories, e.g., into core packages, compilers, or tools. To load additional modules

[30] Easybuild https://docs.easybuild.io

```
$ module load XYZ
```

can be used, where `XYZ` needs to be replaced by the name of the module as it is returned by the `module avail` command. In case a specific module cannot be found, the user can check if it is available somewhere in the `Easybuild` hierarchy by executing:

```
$ module spider XYZ
```

where `XYZ` is the search term. The output of this command will inform the user if such a package is available and if yes, what is necessary to load the corresponding module, i.e., if certain dependencies need to be satisfied. This could, e.g., be the case if a module has been compiled with a different compiler and a change of the compiler environment is necessary. Finally, to get rid of a loaded module `XYZ`, the command:

```
$ module unload XYZ
```

can be executed. To remove all loaded modules the user needs to type

```
$ module purge
```

Further information can be obtained from the `Easybuild` website[31].

### *4.3.2 Available file systems and data transfer*

Most systems at JSC connect to the JUST and mount corresponding GPFS resources via the NFS protocol. Table 8 summarizes the different mount points and their purposes. Note that the folders can be accessed directly via their environment variables `$VAR`, see Table 8.

| Mount point (`$VAR`) | Size | Description |
|---|---|---|
| `$HOME` | 2.8TB | This is the user's home folder (for personal data; available on login and compute nodes). |
| `$SCRATCH` | 9.1PB | This is the compute-projects' standard scratch (temporary storage for applications; available on login and compute nodes). |
| `$PROJECT` | 2.3PB | This is compute-projects' standard directory (for source code, compilation, and executables; available on login and compute nodes). |
| `$FASTDATA` | 9.1PB | A folder suited for applications requiring fast input/output (I/O) - only available upon special request; available on login and compute nodes; protected with snapshots. |

[31] Easybuild https://docs.easybuild.io/en/latest/Introduction.html

| Mount point ($VAR) | Size | Description |
|---|---|---|
| $DATA | 14PB | The data directory of the projects (for storing and sharing data inside a project; available on login nodes). |
| $ARCHIVE | 1.9PB | The data tape backup directory (for long-term storage via the Tivoli Storage Manager (TSM); available on login nodes). |
| $LOCALSCRATCH | | This is the node-local NVMe directory (only available on compute nodes). |

**Table 8: Available file systems on the JUWELS and JURECA system.**

**Please note:** Data in `$SCRATCH` are checked regularity for untouched files and directories. The former are deleted after 90 days and the latter after three days!

Obviously, some of the environment variables `$VAR` are dependent on the compute-time project. To correctly set the variables, the following command can be executed:

```
$ jutil env activate -p project
```

where the `project` needs to be replaced by the id of the compute-time project.

Since outgoing `ssh` connections are not allowed, file transfers to and from JUWELS and JURECA, which use `ssh` as the underlying transport, have to be initiated from the other system. So instead of (replace [`juwels/jureca]` by the corresponding system in the following)

```
[juwels/jureca]$ scp my_file local:
```

the copy needs to be initiated from the local system:

```
local$ scp [juwels/jureca]:my_file .
```

In some cases, it might not be possible to directly transfer files between another system and JUWELS or JURECA. This might be caused by the other system also disallowing outgoing `ssh` connections, or the `ssh` client on the other system is too old and does not support the modern cryptographic algorithms required by JSC's policy. As a workaround, the files have to be transferred to a third system which can make connections to both JUWELS/JURECA and the other system. This can be automated with `scp`:

```
local$ scp -3 other.hpc.example.com:my_file [juwels/jureca]:
```

Another excellent tool to synchronize data is the `rsync` tool. In the following example a file is synchronized from JUWELS/JURECA to the local machine:

```
local$ rsync -av --stats --progress [juwels/jureca]:my_file .
```

The progress is tracked with the `--progress` option and a final summary is provided through the `--stats` option.

### *4.3.3 Software development*

#### Compilers

JURECA has three major compilers available: `GCC`, `Intel`, and `PGI`. The available compiler versions and the corresponding module names can be listed using the module availing and inspecting the section on “Compilers”.

For most applications, the Intel compilers provide the highest performance on the JUWELS and JURECA platforms as a rule of thumb. However, it is recommended to experiment with different compilers and compiler options to find an optimal setting for the specific application. The following Table 9 shows the names of the MPI wrapper procedures for the `Intel` compilers and the names of compilers themselves. The wrappers build up the MPI environment for the compilation task. Therefore, it is recommended to always use the wrappers instead of the compiler drivers themselves.

| Programming language | Wrapper | Intel compiler |
|---|---|---|
| FORTRAN90 | mpif90 | ifort |
| FORTRAN77 | mpif77 | ifort |
| C++ | mpicxx | icpc |
| C | mpicc | icc |

**Table 9: Intel compilers and MPI wrappers.**

#### GPU development

To use the GPUs in JSC’s systems, a GPU-aware toolchain needs to be used. Software compatibility imposes certain restrictions on backend compilers. Therefore, specific compiler versions are needed to enable GPU usage. Software with specific GPU support are marked with a `(g)` at their side when listing modules.

Currently, the only CUDA-aware MPI runtime available on the system is `MVAPICH2`. It is important to note that as of today, this MPI runtime will only work on the GPU nodes. However, equivalent software (without GPU support) has been installed for toolchains (combinations of compiler and MPI runtime) that can be used outside of the GPU nodes.

#### KNL development

To access the software installed for the KNL architecture, i.e., to use the Booster Module of JURECA, the following command has to be executed before loading any other module:

```
$ module load Architecture/KNL
```

In case cross-compilation for the Booster partition is performed on the JURECA login nodes, the `Architecture/KNL` module needs to be loaded as well. In addition, the `module load` command needs to be added to the job script, see the following Sec. 4.3.4.

### *4.3.4 Batch systems*

The systems use the `SLURM` job scheduler for batch system management, see Sec. 2.3.4. `SLURM` offers interactive and batch jobs (scripts submitted into the system).

On JUWELS, jobs need to be allocated to one of the partitions listed in Table 10. The maximum wall-clock time for all production queues (`batch`, `mem192`, `gpus`, and `booster`) is 24 hours. In `no-cont` mode, a project already consumed all its assigned resources, the user can, however, still run computations that are assigned a lower priority. The maximum wall-clock time is in this case, reduced to 6 hours. The default for all queues is one hour in the production queues.

In addition to the partitions listed in Table 10, the `large` and `largebooster` partitions are available for large and full-system jobs. The partitions are open for submission but jobs will only run in selected timeslots. The use of these partitions needs to be coordinated with the user support.

| Name | Description |
|---|---|
| `batch` | normal Cluster partition, where between 1-1,024 nodes can be allocated; if not specified (see `mem192` below) both nodes with 96 and 192 GB of RAM may be in the node set |
| `mem192` | normal Cluster fat node partition, where between 1-64 nodes can be allocated |
| `devel` | development queue on the Cluster, where between 1-8 nodes can be allocated; the max. wall-clock time is reduced to two hours and the default wall-clock time to 30 minutes |
| `gpus` | uses between 1-46 GPU nodes on JUWELS Cluster (4 x NVIDIA V100 GPUs) |
| `develgpus` | development queue using the GPUs on the Cluster (4 x NVIDIA V100), where between 1-2 nodes can be allocated; the max. wall-clock time is reduced to two hours |
| `booster` | Booster partition (4 x NVIDIA A100), where between 1-384 nodes can be allocated |
| `develbooster` | development queue on the Booster (4 x NVIDIA A100), where between 1-4 nodes can be allocated; the max. wall-clock time is reduced to one hour |

**Table 10: Available SLURM partitions on the JUWELS system.**

Table 11 lists the partitions on JURECA. Similar to the JUWELS system, the production partitions on JURECA (`cd-cpu`, `dc-gpu`, `dc-cpu-bigmem`, `booster`) have a maximum wall-

clock time of 24 hours. In `no-cont` mode, 6 hours are possible. The default for all queues is one hour in the production queues. The nodes in the `booster` and `booster-devel` partition are configured with Quadrant Non-Uniform Memory Access (NUMA) Mode with Cache Memory mode.

In addition to the partitions listed in Table 11, the `dc-cpu-large`, `dc-gpu-large` and `booster-large` partitions are available for large and full-module jobs. The partitions are open for submission, but jobs will only run in selected timeslots. Again, the use of these partitions needs to be coordinated with user support.

| Name | Description |
|---|---|
| `dc-cpu` | This is the normal Cluster partition, where between 1-64 nodes can be allocated; if not specified (see `dc-cpu-bigmem` below) both nodes with 512 and 1,024 GB of RAM may be in the node set. |
| `dc-cpu-bigmem` | The normal Cluster fat node partition, where between 1-48 nodes can be allocated. |
| `dc-cpu-devel` | The development queue on the Cluster, where between 1-4 nodes can be allocated; the max. wall-clock time is reduced to two hours and the default wall-clock time to 30 minutes. |
| `dc-gpu` | This uses between 1-24 GPU nodes on JUWELS Cluster (4 x NVIDIA A100). |
| `dc-gpu-devel` | This is the development queue using the GPUs on the Cluster (4 x NVIDIA A100), where between 1-4 nodes can be allocated; the max. wall-clock time is reduced to two hours and the default wall-clock time to 30 minutes. |
| `booster` | The Booster partition (KNLs), where between 1-512 nodes can be allocated. |
| `booster-devel` | The development queue on the Booster (KNLs), where between 1-8 nodes can be allocated; the max. wall-clock time is reduced to 6 hours and the default wall-clock time to 30 minutes. |

**Table 11: Available SLURM partitions on the JURECA system.**

Additional information on partitions on both systems can be obtained by (replace `partition` with the partition name):

```
$ scontrol show partition partition
```

### Interactive jobs

Interactive jobs can be started by first allocating nodes for execution

```
$ salloc –partition=partition --nodes=2 --account=projectid \
  --time=00:30:00
```

In this example, two nodes (`--nodes=2`) on the partition `partition` (replace by a suitable partition name) is requested for `30` minutes on budget `projected` (replace by your project id). An executable that then uses all the requested resources can be run by (replace `program` by your executable), e.g.,

```
$ srun --ntasks=4 --ntasks-per-node=2 --cpus-per-task=8 ./program
```

where `--ntasks` defines the total number of MPI ranks, `--ntasks-per-node` the number of MPI ranks per node, and `--cpu-per-task` the number of hardware threads.

If a shell on the first allocated compute nodes is needed, a remote shell from within the `salloc` session can be started and connected to a pseudo terminal using:

```
$ srun --cpu_bind=none --nodes=2 --pty /bin/bash -i
```

The option `--cpu_bind=none` is used to disable CPU binding for the spawned shell. In order to execute MPI applications, `srun` can be used again from the remote shell. To support `X11` forwarding the `--forward-x` option to `srun` is available. `X11` forwarding is required for users who want to use applications or tools with a GUI support.

### Batch jobs

Jobs can also be submitted via a job script that holds all necessary information. Therefore, a job file (here in this example `job.slurm`) needs to be created. For the above example the job script needs to contain:

```
#!/bin/bash
#SBATCH --partition=partition
#SBATCH -A projectid
#SBATCH -N 4
#SBATCH -n 8
#SBATCH -o /p/project/projectid/user/hello_cluster-%j.out
#SBATCH -e /p/project/projectid/user/hello_cluster-%j.err
#SBATCH --time=00:10:00
srun ./program
```

Output and error files are placed in the files provided with the `-o` and `-e` options. The options `-A`, `-N`, `-n` correspond to the `--account`, `--ntasks`, and `--ntasks-per-node` options see above). With this in mind, jobs of any kind can be submitted using MPI parallelization solely with a variable number of MPI ranks per node, and an additional set of threads per MPI rank. For OpenMP, the number of OpenMP threads per MPI rank additionally needs to be exported inside the job script:

```
...
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
srun ./program
```

The job can then be submitted via

```
$ sbatch job.slurm
```

This will return a job id. The status of the job can then be checked with (replace `jobid` by the id of the job):

```
$ squeue --job jobid
```

Jobs can be canceled with

```
$ scancel jobid
```

**GPU computing on JUWELS and JURECA**

The GPU nodes in the JUWELS Booster feature four NVIDIA A100 GPUs. The nodes are accessible in the Booster partition. To use this partition, the argument `-p booster` (or `--partition booster`) must be provided to `sbatch` or `salloc`. Also, the number of requested GPUs must be specified using the `--gres=gpu:X` argument with `X` in the range one to four. Similarly, the GPUs on the JUWELS Cluster can be used with the arguments `-p gpus` (or `--partition gpus)`, and with the `--gres=gpu:X` argument.

The GPU nodes in the JURECA DC module feature four NVIDIA A100 GPUs. The nodes are accessible in the `dc-cpu` partition. To access this partition, the argument `-p dc-gpu` (or `--partition dc-gpu`) needs to be provided.

For independent instances (job steps) of a GPU program running on a JUWELS Cluster or Booster node using one CPU thread and one GPU device each, the program is pinned to certain CPU cores in the job script (here `18`, `6`, `42`, and `30`):

```
...
#SBATCH --gres=gpu:4
srun --exclusive -n 1 --gres=gpu:1 --cpu-bind=map_cpu:18 ./gpu-prog &
srun --exclusive -n 1 --gres=gpu:1 --cpu-bind=map_cpu:6  ./gpu-prog &
srun --exclusive -n 1 --gres=gpu:1 --cpu-bind=map_cpu:42 ./gpu-prog &
srun --exclusive -n 1 --gres=gpu:1 --cpu-bind=map_cpu:30 ./gpu-prog &
wait
```

**Heterogeneous jobs**

To make use of the MSA features on JUWELS and JURECA, jobs can be specified that make use of multiple types of partitions, or in other words, of multiple architectures. This way, a heterogeneous job can, e.g., be spawned across multiple supercomputing modules.

The syntax of the interactive and non-interactive submission mechanisms – `salloc` and `srun` – has been extended to the user to specify individual options for the different job components. For `srun`, the command line sequence is partitioned into several blocks with the colon `:` acting as the separator. The resulting heterogeneous job will have as many job components as there were blocks of command-line arguments. The first block of arguments contains the job options of the first job component and common job options that will apply to all other components. The second block contains options for the second job component, and so on. The abstract syntax is as follows:

```
$ salloc <options 0 + common> : <options 1> [ : <options 2>... ]
```

The following invocation of `salloc` submits an interactive heterogeneous job that consists of two components, the first requesting one node from the `partition_a` partition, the second requesting 16 nodes from the `partition_b` partition:

```
$ salloc -A budget -p partition_a -N 1 : -p partition_b -N 16
```

Submitting non-interactive heterogeneous jobs through `sbatch` works similarly, but the syntax for separating blocks of options in a batch script is slightly different. Instead of the colon `:`, batch scripts use the usual directive `#SBATCH`, followed by the word `packjob` as a separator:

```
#!/bin/bash
#SBATCH <options 0 + common>
#SBATCH packjob
#SBATCH <options 1>
[
#SBATCH packjob
#SBATCH <options 2>...
]
```

To submit a non-interactive heterogeneous job with the same setup as the interactive job above, the job script would read

```
#!/bin/bash
#SBATCH -A budget -p partition_a -N 1
#SBATCH packjob
#SBATCH -p partition_b -N 16
...
```

Further job options can also be provided in the `sbatch` command line and also by mixing options in the `sbatch` command line and the job script. Again, the colon `:` acts as the separator of blocks of command-line arguments. For example, to specify that particular job components should always run on certain partitions, they could be specified in the job script, while the number of nodes is left to be specified on the command line. The following batch script, submitted via

```
$ sbatch -N 1 : -N 16 <batch script>
```

results in the same heterogeneous job as the previous two examples.

```
#!/bin/bash
#SBATCH -A budget -p partition_a
#SBATCH packjob
#SBATCH -p partition_b
...
```

As with homogeneous jobs, applications are launched inside a heterogeneous job using `srun`. Like `salloc` and `sbatch`, `srun` can be used to specify different options and also commands to run for different components through blocks of command line arguments separated by the colon `:`.

```
$ srun --ntasks-per-node 24 ./prog1 : --ntasks-per-node 1 ./prog2
```

The option `--pack-group=<expr>` can be used to explicitly assign a block of command line arguments to a job component. It takes as its argument `<expr>` either a single job component index in the range 0 ... n - 1 where n is the number of job components, or a range of indices like 1-3 or a comma separated list of both indices and ranges like 1,3-5. The following invocation of `srun` runs the same application `./prog` in components 0 and 2 of a three component heterogeneous job, leaving component 1 idle:

```
$ srun --pack-group=0,2 ./prog
```

The same application `./prog` can be run in all three job components using:

```
$ srun --pack-group=0-2 ./prog
```

In case a uniform architecture is used, modules can be loaded as before

```
#!/bin/bash -x
#SBATCH ...
module load [...]
srun ./prog1 : ./prog2
```

In case a non-uniform hardware is used and additional hardware-specific libraries etc. need to be loaded, the job file needs to be modified to read, e.g.,

```
...
srun --account=<budget account> \
  --partition=<batch, ...> xenv -L intel-para IMB-1 : \
  --partition=<knl, ...> xenv -L Architecture/KNL -L intel-para IMB-1
```

# 5 RUDENS at Riga Technical University

The majority of HPC systems, including the one available at RTU, are designed as computing clusters, which consist of numerous separate servers interconnected by a fast computer network, e.g., InfiniBand. HPC cluster can be used for both parallel computing, providing parallel execution of a large job, and distributed computing, performing several independent tasks on separate servers or processors.

In this section, the system RUDENS, installed at the RTU HPC Centre is presented. Technical support is provided to all HPC cluster users. The support includes assistance in preparing jobs, installing application software, briefing on the work with the cluster, and consulting. In some areas such as computational chemistry, scientific support is also provided by helping to choose the most appropriate simulation tool for the problem or to create a simulation model. The HPC user support can be reached via hpc@rtu.lv. Security incidents should be reported to the same address.

In the following, Sec. 5.1 gives detail on RUDENS' hardware specifications. Section 5.2 provides the reader with information on how to access the system before a usage guide is presented in Sec. 5.3.

## 5.1 Hardware specifications

RTU's HPC cluster RUDENS, shown in Figure 10, consists of 54 compute nodes for job execution and one head node, which performs cluster management operations. All nodes are inter-connected with fast InfiniBand network. Each compute node is equipped with two `x86_64` architecture CPUs. Some of these nodes are additionally equipped with 2 or 4 NVIDIA Tesla GPUs. The cluster architecture is heterogeneous, combining nodes of different generations and technical parameters.

Several Network-Attached Storage (NAS) systems are available for storing user data with a total capacity of ~800 TB. For tasks with intensive I/O, a special NVMe disk array with a BeeGFS parallel file system[32] is available.

Users connect to a separate login node, which provides a work environment and tools for running jobs on the cluster. Job management is provided by the Terascale Open-source Resource and QUEue Manager (`TORQUE`) and `Moab` tools[33].

The following gives an overview of the general system parameters:

- 54 computing nodes
- 1,200+ CPU cores
- 8 TB RAM (up to 1.5 TB per process)
- 16 NVIDIA Tesla GPUs
- 132 TFLOPs overall performance (62 TFLOPs x86 + 70 TFLOPs GPU)
- 800 TB data storage
- 10 Gb/s access network
- 40-100 Gb/s InfiniBand interconnection

[32] BeeGFS https://www.beegfs.io
[33] TORQUE and Moab tools http://docs.adaptivecomputing.com/

There exists a 10 Gb/s connection to the Latvian Academic computer network, European Academic Computer Network GEANT, Genomics Data Network, and commercial Internet providers in Latvia. More details on the node configurations and the data storage systems can be found in the subsequent sections.

**Figure 10: The RUDENS system installed at RTU HPC.**

### *5.1.1 Node configuration of RUDENS*

The cluster is composed of different login and computing nodes which are listed in the following:

**Login nodes**

- **`ui-1.hpc.rtu.lv`** (primary): 2 x Intel Xeon CPU E5-2680 v3, clocked at 2.50 GHz (total 24 cores), 128 GB RAM, 10 Gb/s ethernet access network, 56 Gb/s InfiniBand FDR network
- **`ui-2.hpc.rtu.lv`** (backup): 2 x Intel Xeon CPU E5630, clocked at 2.53 GHz (total 8 cores), 24 GB of DDR3 RAM with a clocking of 1066 MHz ECC, 1 Gb/s ethernet access network, 40 Gb/s InfiniBand Quad Data Rate (QDR) network

**CPU computing nodes**

- **10 nodes** Dell EMC PowerEdge R640 (**`wn02-wn10`**)**:** 2 x Intel Xeon Gold 6154 CPU, clocked at 3.00 GHz (total 36 cores), 384 GB of DDR4 RAM with a clocking of 2666 MHz ECC, 1 TB local SSD, InfiniBand EDR 100 Gb/s network
- **11 nodes** Dell PowerEdge R630 **(`wn45-wn55`)**: 2 x Intel(R) Xeon(R) CPU E5-2680 v3, clocked at 2.50 GHz (total 24 cores), 128 GB DDR4 RAM with a clocking of 2133 MHz ECC, 200 GB SSD, InfiniBand FDR 56 Gb/s network
- **24 nodes** *T-Platforms T-Blade2* (**`wn12-wn34`**, **`wn42`**): 2 x Intel(R) Xeon(R) CPU X5670, clocked at 2.93 GHz (total 12 cores), 12 GB of DDR3 RAM with a clocking of 1333 MHz ECC, InfiniBand QDR 40 Gb/s network
- **2 nodes** T-Platforms T-Blade2 (**`wn35`**, **`wn37`**): 2 x Intel Xeon CPU E5630, clocked at 2.53 GHz (total 8 cores), 24 GB of DDR3 RAM with a clocking of 1066 MHz ECC, InfiniBand QDR 40 Gb/s network

**High memory nodes**

- **1 node** Dell PowerEdge R940 (**`wn01`**): 4 x Intel Xeon Gold 6140 CPU, clocked at 2.30 GHz (total 72 cores), 1.5 TB of DDR4 RAM with a clocking of 2666 MHz ECC, 1 TB SSD, InfiniBand EDR 100 Gb/s network

**GPU computing nodes**

- **2 GPU nodes** Dell EMC PowerEdge C4140 (**`wn59-wn60`**): 2 x Intel Xeon Gold 6130 CPU, clocked at 2.10 GHz (total 32 cores), **4 x NVIDIA Tesla V100 GPU**, 16 GB HBM2, 5120 CUDA cores, 192 GB DDR4 RAM at 2666 MHz ECC, 240 GB SSD, InfiniBand EDR 100 Gb/s network
- **4 GPU nodes** Dell PowerEdge R730 (**`wn43, wn56-wn58`**): 2 x Intel(R) Xeon(R) CPU E5-2680 v3, clocked at 2.50 GHz (total 24 cores), **2 x NVIDIA Tesla K40 GPU**, 12 GB GDDR5, 2888 CUDA cores, 128 GB of DDR4 RAM at 2133 MHz ECC, 200 GB SSD, InfiniBand FDR 56 Gb/s network

**Cluster management software**

- Operating system (OS): Linux CentOS 7
- Job Management: `TORQUE 6.1.1.1` / `Moab 9.1.1`
- Accounting: `Moab` Accounting Manager
- OS image provisioning: *xCAT*

The different nodes available and their features are shown in Table 12.

| Name | Type | OS | RAM | CPU | CPU cores | GPUs | Local disk | `qsub` features |
|---|---|---|---|---|---|---|---|---|
| `RUDENS` | cluster head | CentOS 6 | 32 GB | E5-2620 v3 | 20 | - | - | - |
| `ui-1` | login | CentOS 7 | 64 GB | E5-2650 v3 | 20 | - | `/scratch` | - |
| `ui-2` | login | CentOS 7 | 24 GB | E5630 | 8 | - | - | - |
| `wn01` | compute-smp | CentOS 7 | 1.48 TB | Gold 6140 | 72 | - | `/scratch` | `dell vasara highmem centos7 largescratch` |
| `wn02-wn11` | compute | CentOS 7 | 384 GB | Gold 6154 | 36 | - | `/scratch` | `dell vasara centos7 largescratch gold6154` |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| wn12-wn34, wn42 | compute | CentOS 6 | 12 GB | X5670 | 12 | - | - | tb2 inf centos6 |
| wn35-wn39 | compute-gpu | CentOS 7 | 24 GB | E5630 | 8 | 2 | - | tb2-8 gpu centos7 |
| wn45-wn55 | compute | CentOS 7 | 128 GB | E5-2680 v3 | 24 | - | /scratch | dell RUDENS centos7 |
| wn43, wn56-wn58 | compute-gpu | CentOS 7 | 128 GB | E5-2680 v3 | 24 | 2 | /scratch | dell RUDENS gpu k40 centos7 |
| wn59-wn60 | compute-gpu | CentOS 7 | 192 GB | Gold 6130 | 32 | 4 | /scratch | vasara vasara-32 gpu v100 centos7 |

**Table 12: Different nodes available in RUDENS.**

The different data storage systems are given in the following.

**NFS network data storage** (238 TB, `/home`)

- 8 nodes EMC Isilon x200
- Distributive file system OneFS
- Client connection with NFS protocol
- 40 Gb/s InfiniBand connection to the cluster
- 238 TB SATA SDD + SATA spinning disks

**BeeGFS network data storage #2** (430 TB, `/home_beegfs`; `/scratch_beegfs`)

- 2 nodes SuperStorage 6029P
- Parallel file system BeeGFS
- 100Gbit/s Infiniband network
- 30 TB NVMe fast disk pool
- 400 TB NL SAS disk pool

**BeeGFS network data storage #1** (104 TB, `/mnt/beegfs`)

- 2 nodes: SuperStorage 6029P and Lenovo x3650 M5
- Parallel file system BeeGFS
- 10 Gb/s Infiniband connection to the cluster
- 100 TB SATA HDD disk pool

**Scratch disks** (15 TB in total)

- Compute nodes are equipped with local SDD disks with a capacity of up to 1 TB.

## 5.2 Accessing the system

To receive access credentials, the applicant must fill in the application form[34]. The CoE RAISE project needs to be mentioned in the application. After receiving the application, the HPC Centre will contact the applicant and will provide additional information on the following steps. The user will have to sign a User Agreement. The terms of use are available online[35].

## 5.3 Using the system

This section provides an overview of how to use the RUDENS system at RTU HPC Centre. Section 5.3.1 gives details on the available user interfaces. Subsequently, Sec. 5.3.2 presents a getting-started guide, before Sec. 5.3.3 provides information on the available software. Section 5.3.4 dives deeper into the job management tools. Using GPUs and CUDA on the system is explained in Sec. 5.3.5 and running ML applications in Sec. 5.3.6. Finally, Sec. 5.3.7 provides information on the usage accounting.

### *5.3.1 User interfaces*

There are several options to work on the cluster:

**Command line.** This is a traditional way of accessing the cluster. A user connects over the internet to a remote terminal, usually through an `ssh` protocol. Commands can be executed in a text mode. It is possible to edit and compile a code, place a job (simulation) in a submission queue, and monitor the job's course of execution. Graphical tools can also be called from the command line. Instructions in this guide are mainly intended for command-line access.

**Graphical user interface (GUI) of application software.** Some software packages, e.g., Schrodinger, MATLAB, etc., have built-in support for parallel computing on a cluster. Users can either perform a simulation locally or on a cluster, without leaving the regular software environment. In case such software is used, the cluster can be utilized by opening the graphical interface either locally on the personal computer of the user or remotely on the cluster login node.

**Web interface.** Access to the cluster using an `HTTPS` protocol can be obtained by opening a graphical interface (portal) via an internet browser.

- **Moab Viewpoint** is a graphical interface for submitting and managing jobs similar to using command line tools but a graphical environment. To access this method, the support should be contacted (hpc@rtu.lv).
- **Galaxy**[36] is an open platform for supporting data intensive research with focus on biomedical data, e.g., for genome, transcriptome, or proteome analyses. The analyses can be performed using graphical tools and could hence easily be performed by non-bioinformaticians. The RTU HPC Galaxy instance is available via a webpage[37]. To gain access to Galaxy, an email should be written to the support (hpc@rtu.lv). Automatic registration is also possible with the RTU HPC cluster credentials. Any requests regarding tools or reference datasets can also be sent to the support.

[34] Application form RUDENS https://docs.google.com/forms/d/e/1FAIpQLSemd1JlJB2lyW0Fal6OA3MM7cmxpgh0GQt145lrzmqqIFQIeA/viewform
[35] RUDENS terms of use https://hpc.rtu.lv/hpc/hpc-services/using-hpc/?lang=en
[36] Galaxy http://galaxyproject.org/
[37] RTU Galaxy https://galaxy.hpc.rtu.lv

Figure 11 shows an example quality control report in Galaxy using FastQC, a quality control tool for high throughput sequence data.

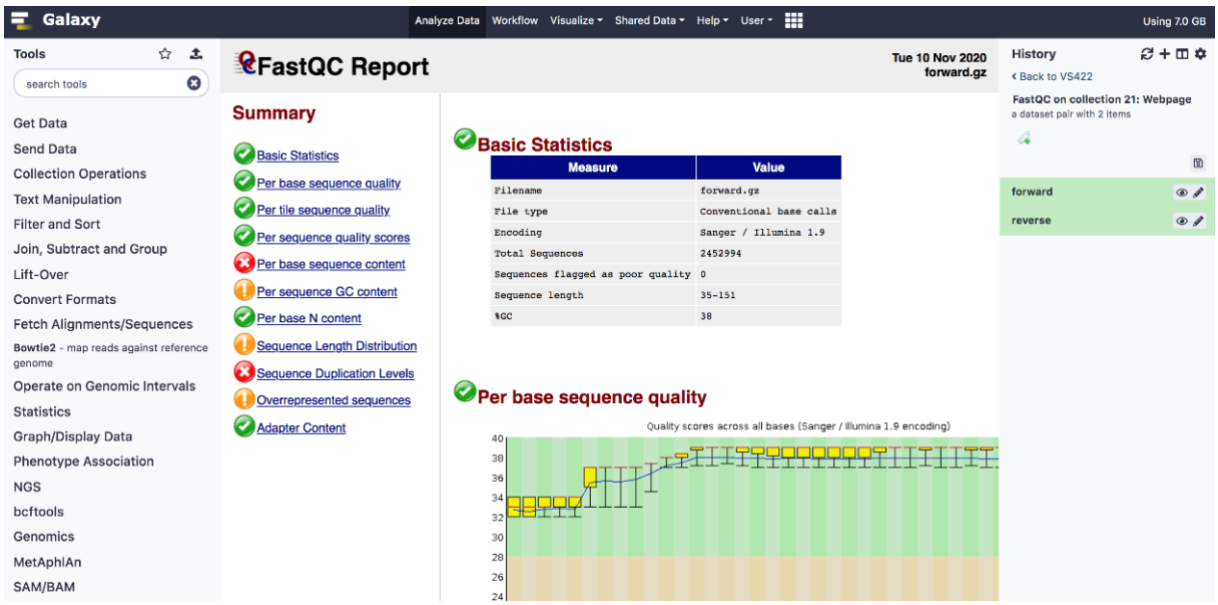



**Figure 11: Galaxy platform on RUDENS.**

### *5.3.2 Getting started on the cluster*

**Accessing the cluster**

To work with the cluster, a dedicated server (login node) with CentOS 7 Linux operating system and job management tools installed can be used. After a connection is established, the user is able to use the command line with UNIX commands, edit and compile code, submit a job (simulation), or to monitor its execution on the computing resources. From the command line, the user can also call graphical tools/windows as described below.

**Note:** The users are asked not to use the login nodes for operations with high resource-consumption. The login nodes are meant only for copying files to or from the cluster, preparing jobs, submitting jobs to a queue, and monitoring results. Testing a short (<5 min.) job or compiling a code on a few CPU cores is acceptable. Running code or simulations on the login node (also opening a software GUI) does not automatically ensure that the job will run on the compute nodes. This requires to use the job management tools described in Sec. 5.3.4.

The command-line access parameters are:

- primary login node: `ui-1.hpc.rtu.lv`
- backup login node: `ui-2.hpc.rtu.lv`
- protocol: `ssh`
- port: 22
- The user authenticates against the system via the login name and password as received when registering for the cluster access.
- Access is ensured from any IP address.

Connections via `ssh` can be established with the following tools:

- On a Linux or MacOS system, the user can connect to the cluster with `ssh` by opening the command line (terminal) and running the following command:

```
$ ssh <username>@ui-1.hpc.rtu.lv
```

- For Windows operating system, PuTTY[38] can be used.

When using the cluster through an `ssh` client such as PuTTY, not only the command line terminal but also the graphical tools may be used. `ssh` ensures `X11` forwarding from the cluster to the user's computer. To use X11 forwarding, the `-X` parameter can be used:

```
$ ssh -X <username>@ui-1.hpc.rtu.lv
```

To use this function on a Windows system, additional preparation on the user's personal computer is required:

1. Installation of an X Windows server, e.g.,VcxSrv[39].
2. The server needs to run in the background as a service. The first time the server is started, the user may be prompted to change the settings. The default values should be kept.
3. In addition, `X11` forwarding needs to be enabled in PuTTY's configuration.

The following example opens a MATLAB GUI from the command line. To start the MATLAB GUI on the cluster login node (`ui-1.hpc.rtu.lv`), the user first needs to connect to the cluster command line by using the graphical session forwarding described above.

On the login node, the following commands needs to be executed:

```
$ module load matlab/R2018a
$ matlab &
```

A program window will pop up on the user's computer, but all operations will run remotely on the cluster. The MATLAB interface provides the necessary tools for queuing and executing a task on computing nodes.

**User workspace**

Each user is provided with a prepared workspace, intended for files related to the jobs. By logging into the system using the command-line interface, the user is automatically directed to the home directory `/home/username`. This directory is shared between all cluster nodes using NFS or BeeGFS, i.e., there is no need to copy files from the login node to compute nodes, where the jobs are executed, as well as between the compute nodes in parallel job setups, see Figure 12.

[38] PuTTY http://www.putty.org/
[39] VcxSrv https://sourceforge.net/projects/vcxsrv/

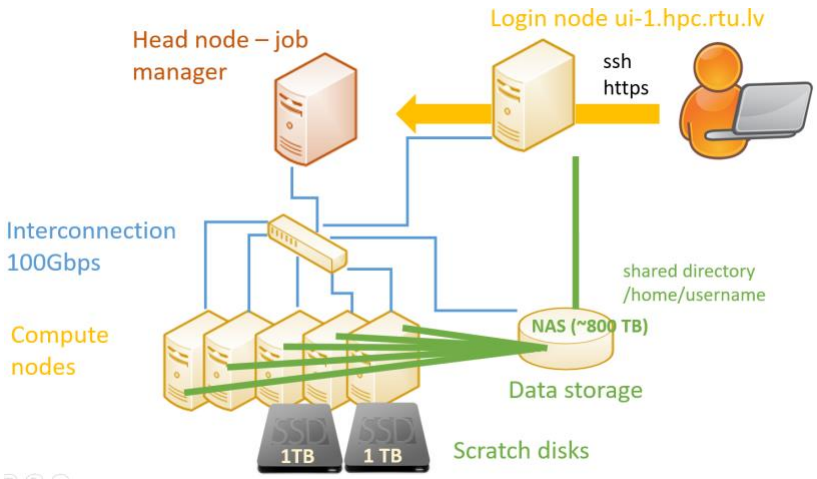



**Figure 12: System and user's workspace on the RUDENS system.**

As mentioned above, the users can install/compile software in their workspace if the installation process does not require administrator (`root`) permissions.

To copy files from the user's computer to the cluster login node, Windows users can use tools such as WinScp[40] or the FAR file manager[41].

On a Mac or Linux operating system, the `scp` command or any convenient graphical tool can be used. An example of how to copy a file from a Linux client system to the user's home folder could be:

```
$ scp -r my.file username@ui-1.hpc.rtu.lv:
```

### *5.3.3 Software installed on the cluster*

The full list of scientific software and tools installed on the HPC cluster RUDENS is available online[42]. Upon request, additional software or other versions may be installed. CentOS, EPEL, and OpenHPC[43] repositories are used on the cluster. Tools available through the OpenHPC repository can be found on GitHub[44].

Users can also install/compile the software in their user area in case the installation process does not require administrator (`root`) permissions.

The cluster directory `/opt/exp_soft/user_info` contains examples of software usage and job batch scripts.

#### Using software modules

Users can use modules (Environment Modules[45]) to prepare their environment for running different software, compilers, and libraries available on the cluster. A software module is loaded with the command:

[40] WinScp https://winscp.net/eng/download.php
[41] Far file manager https://www.farmanager.com
[42] RUDENS software list https://hpc.rtu.lv/hpc-manual/software-installed-on-the-cluster/?lang=en
[43] OpenHPC https://openhpc.community/
[44] OpenHPC tools https://github.com/openhpc/ohpc/wiki/Component-List-v1.3.8
[45] Environment Modules http://modules.sourceforge.net/

```
$ module load <module_name>
```

It sets the path (`PATH`) to executable files, libraries (`LD_LIBRARY_PATH`), and other required environment variables for a particular software.

To get a list of all possible modules (software and tools pre-installed on the cluster), the following command can be executed:

```
$ module avail
```

It should be noted that to use the tools (packages) included within Linux distributions, it is usually not required to load a module.

The module must be loaded on the node where the software is started. If a job is executed on a compute node, the corresponding module must be loaded on the node as well or should be passed from a login node with job management commands.

To get a list of loaded modules

```
$ module list
```

can be used. To unload a module

```
$ module unload [module_name]
```

can be utilized. This is similar to using the `Easybuild` system as presented in Sec. 4.3.1.

**Private modules**

It is also possible to create own modules from user-compiled software. The following steps describe briefly what needs to be done in this case

- A new directory needs to be created:

  ```
  $ mkdir /home/<username>/privatemodules
  ```

- In this directory, the own module can be created. Examples can be found in the `/opt/exp_soft/modulefiles` directory.

- The private module directory needs to be activated:

  ```
  $ module load use.own
  ```

  This command can be put in the `~/.bashrc` file,

- Finally, the own module needs to be loaded:

  ```
  $ module load <module_name>
  ```

### *5.3.4 Job management*

The cluster uses a dedicated job management system that helps to organize users' jobs. When a user requests specific resources for a job, e.g., a particular number of CPU cores or a certain amount of memory, the management system finds and allocates resources and ensures exclusive access to them, i.e., it schedules the job in such a way that it does not overlap with other users' jobs. Virtual queues are used to fairly distribute cluster resources between jobs of many users. A job execution requires the following steps:

1. Logging into the cluster;
2. Queuing a job;
3. Executing and monitoring a job;
4. Receiving the results.

The job management system used at RTU is `TORQUE` / `Moab`. `TORQUE` is a basic batch system, and `Moab` provides higher-level job scheduling and cluster management functionalities.

The maximum execution time of one job (simulation) is two weeks. After this period, the job will be automatically cancelled. If more time is needed for a job, the user can make a separate agreement with the HPC Centre.

If possible, the execution of jobs locally on the login nodes should be avoided. They should only be used to submit a job to a queue, to compile programs, and to test short jobs.

The user should try to indicate the necessary wall clock time such that other users can forecast their queueing time.

If a job requires intensive I/O, it is advised to use a local SSD disk instead of the network directory (`/home/*` or `/home_beegfs/*`). The local disk is mounted to the directory `/scratch`. When the job is completed, the files created in this directory should be removed by the user manually.

The user should furthermore evaluate the performance of parallel (MPI) jobs by using a various number of nodes/cores to select the most suitable one for the job. A larger number of cores does not always ensure faster execution of a job.

**Queueing a Job**

Before a job is executed on a compute node, it is placed in a virtual queue. The queue organizes resource allocation in a multi-user environment, where the number of jobs and their requirements may exceed the number of available CPU or memory resources. When resources become available, usually the job waiting in the queue the longest will be executed next. Users do not have to monitor the resource availability. Job movement in the queue and job execution are performed automatically. If there are no waiting jobs in a queue, i.e., the resources are available, the job is started immediately.

Jobs are queued using special `TORQUE` / `Moab` tools. A detailed documentation is available online[46].

[46] TORQUE documentation http://docs.adaptivecomputing.com/torque/6-1-2/adminGuide/torque.htm

The command for submitting a simple job via a batch script is:

```
$ qsub test.sh
```

Here, `test.sh` is a bash script containing commands to execute when the job is pushed to a compute node. It ensures the execution of a batch job without requiring user interactions. Examples of batch scripts launching various software applications on RUDENS and other useful information can be found in the directory `/opt/exp_soft/user_info`.

Alternatively, an interactive job can be submitted. The interactive mode is convenient for testing and debugging jobs or in case graphical tools are used. An interactive job can be submitted via:

```
$ qsub -I
```

This will open a remote terminal on a compute node, where the user can execute the needed commands on the command line. The command is similar to `ssh wn[XX]`, with the difference that resources are exclusively reserved for the user.

In case a graphical session is required, the `-X` parameter can be added to the submission command:

```
$ qsub -X -I
```

For more information on graphics session forwarding, the reader is referred to RTU's user interface documentation[47].

**Job requirements**

Different job parameters and requirements such as the name of the queue or the wall clock time can be specified by the user. This information is used by `TORQUE` / `Moab` to find the most suitable resources for the job.

The following example shows how to submit a job with name `my_job` with a wall clock time of `30` seconds to the queue with name `fast`:

```
$ qsub -N my_job -q fast -l walltime=00:00:30 test.sh
```

It is also possible to specify these requirements at the beginning of the job script:

```
#!/bin/bash
#PBS -N my_job                  ### Job name
#PBS -l walltime=00:00:30       ### Expected job maximum duration
#PBS -l nodes=1:ppn=1           ### Computing resources needed
#PBS -q fast                    ### Queue
#PBS -j oe                      ### Combine standard output and error in
                                ### the same file
```

[47] RTU user interface documentation https://hpc.rtu.lv/connecting-to-the-cluster-manual/?lang=en

The requirements can be entered on the command line and/or in the job script. The highest priority is, however, given to the requirements provided on the command line in case they are specified repeatedly.

To request specific compute resources, the `qsub -l` command is used:

- number of cores (12):

  `-l procs=12`

- number of nodes and cores (2 node each with 12 cores - this notation should be used for MPI jobs):

  `-l nodes=2:ppn=12`

- particular node (may result in a longer queueing time):

  `-l nodes=wn44:ppn=12`

  `-l nodes=wn44:ppn=12+wn46:ppn=12`

- use nodes only from a list:

  `-W x=HOSTLIST:wn02,wn03,wn04,wn05,wn06,wn07,wn08,wn09,wn10,wn11`

- number of GPUs (2):

  `-l nodes=1:ppn=12:gpus=2`

- required amount of memory:

  `-l nodes=1:ppn=12,pmem=1g` (memory per core (processor))
  `-l nodes=1:ppn=12,mem=12g` (total memory for a job)

- Request compute nodes with particular features – an example is shown in Figure 13. Features are usually used on clusters with non-homogeneous architecture. For example, to guarantee that the job is executed on the latest generation of nodes - those with 36 CPU cores per node – the following option needs to be specified.

  `-l feature=vasara`

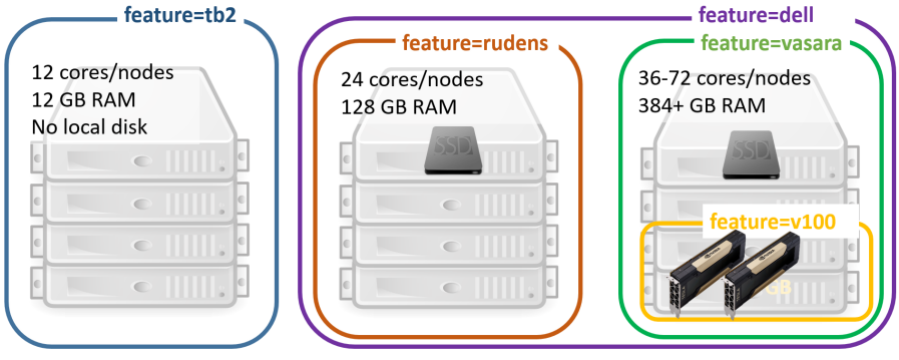



**Figure 13: Example of available nodes at RUDENS.**

For full list of node names and features, the reader is referred to Table 12 in Sec. 5.1.1.

### Parallel (MPI) job execution

A parallel job is distributed on several cores of a node and/or on multiple cluster nodes with the MPI protocol. The following example queues a parallel job requiring 24 cores (2 nodes × 12 cores in each node):

```
$ qsub -l nodes=2:ppn=12 run_mpi.sh
```

It is advised to use the pre-compiled OpenMPI versions with `TORQUE` support. The corresponding module can be loaded via:

```
$ module load mpi/openmpi-<version>
```

MPI job examples can be found in the cluster directory /opt/exp_soft/users_info/mpi.

### Job variables

It is convenient to use variables in a job script that are set automatically when the job arrives on a compute node. The following list mentions the most important variables:

- `$PBS_O_WORKDIR` – the directory from which the job was submitted
- `$PBS_NODEFILE` – list of the nodes reserved for the job
- `$PBS_GPUFILE` – list of GPUs reserved for the task
- `$PBS_NUM_NODES` – number of nodes reserved for the job
- `$PBS_NUM_PPN` – reserved number of cores for each node
- `$PBS_NP` – the total number of cores reserved for the job
- `$PBS_JOBID` – job identifier

For example, to go to the directory from which the job was submitted, the subsequent command can be used:

```
$ cd $PBS_O_WORKDIR
```

To get a list of all possible variables, an interactive job can be started with `qsub -I` and the environment variables can be checked:

```
$ env | grep PBS
```

### Cancelling a job

Running or waiting jobs can be cancelled by executing:

```
$ qdel <job_id>
```

or by using

```
canceljob <job_id>
```

where <job_id> is the unique identifier of a job. To stop all user's jobs, the user can do:

```
$ qdel 'all'
```

### Job queues

There exist several queues on the HPC cluster RUDENS, which differ by the job duration and by the amount of available resources:

- batch – This is the default queue, which is suited for tasks without specific requirements. The maximum job duration is 96 hours.
- fast – This queue for short jobs with a run time of up to 8 h with higher queue priority. It is well suited for interactive and testing jobs. Only a single job per user can be in running state at a time.
- long – This queue is intended for time-consuming tasks with a run time of up to 2 weeks.
- highmem – This queue gives access to high memory nodes, which have a maximum of 1.5 TB per process. The queue is available only upon agreement by the HPC Centre.

The queue can be selected with the -q <queue name> parameter. If the queue is not specified, the job will be placed in the default queue batch.

### Monitoring queues, nodes, and jobs

The command to check the status of submitted jobs is:

```
$ qstat
```

Jobs can be either running (R), completed (C), or queued (Q).

To see all running jobs queued or running on cluster (for all users) the command:

```
$ showq
```

can be used. The command to list the available compute resources is:

```
$ showbf
```

To obtain information on the node availability, the following command is helpful:

```
$ nodes
```

Details on the job execution as well as the reasons for the job being held from execution in the queue can be obtained via:

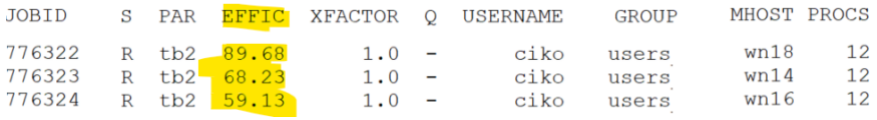
```
$ checkjob job_id -vvv
```

### Job efficiency

It is useful to track the resources that a job actually consumes. The reservation of multiple CPU cores or nodes does not yet guarantee that the job will efficiently use all allocated resources. Information on the CPU usage efficiency for all user's jobs in execution can be obtained via (the command should be executed on login node):

```
$ showq -r -u <username>
```

An example output is given in Figure 14.



| JOBID | S | PAR | EFFIC | XFACTOR | Q | USERNAME | GROUP | MHOST | PROCS |
|---|---|---|---|---|---|---|---|---|---|
| 776322 | R | tb2 | 89.68 | 1.0 | - | ciko | users | wn18 | 12 |
| 776323 | R | tb2 | 68.23 | 1.0 | - | ciko | users | wn14 | 12 |
| 776324 | R | tb2 | 59.13 | 1.0 | - | ciko | users | wn16 | 12 |

**Figure 14: CPU usage efficiency on RUDENS.**

Here, the `EFFIC` column shows the average ratio between the actual processor time and reserved one in %. The following rules of thumb allow to classify the efficiency of a running job:

- 70-100% – Optimal usage of resources;
- < 50% – The job uses allocated resources inefficiently, i.e., not all CPU cores are loaded or the job is mainly waiting for slow I/O operations. In this case it is recommended to reduce the number of parallel processes;
- > 110% – Indicates that the job uses more CPU resources than reserved. This is not recommended.

To obtain information of the local resources consumed by a job on a compute node, the following has to be done:

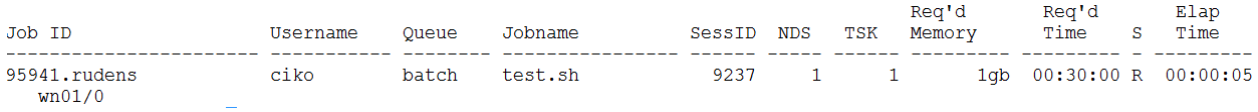
1. Identification of the node the job is running on

   ```
   $ qstat -n job_id
   ```

   or

   ```
   $ qstat -n job_ showq -r -u <username>
   ```

   Figure 15 illustrates the resuls of such a query. The job is running on compute node `wn01`.



| Job ID | Username | Queue | Jobname | SessID | NDS | TSK | Req'd Memory | Req'd Time | S | Elap Time |
|---|---|---|---|---|---|---|---|---|---|---|
| 95941.rudens<br>wn01/0 | ciko | batch | test.sh | 9237 | 1 | 1 | 1gb | 00:30:00 | R | 00:00:05 |

**Figure 15: Job status on RUDENS.**

2. Establishment of an ssh connection to the node wn01:

   ```
   $ ssh wn01
   ```

Usage of the available Linux tools `htop`, `dstat`, `nvidia-smi`, `iostat`, `nfsstat` for monitoring. For example, the efficiency of CPU usage with command `htop`, is shown in Figure 16.

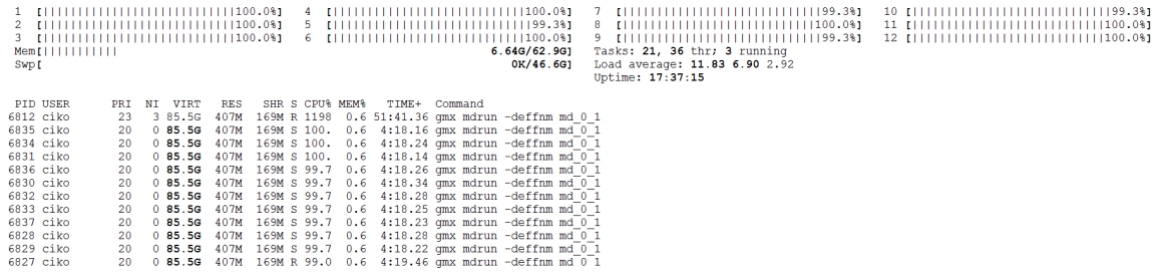



**Figure 16: Using htop at RUDENS.**

To track the GPU usage efficiency, the command `nvidia-smi` can be used. An exemplary output is shown in Figure 17.

```
[ciko@node10 2016.1]$ nvidia-smi
Tue Jul 24 10:24:10 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 396.26                 Driver Version: 396.26                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K20m          On   | 00000000:04:00.0 Off |                  Off |
| N/A   38C    P0    89W / 225W |     81MiB /  5062MiB |     67%   E. Process |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K20m          On   | 00000000:42:00.0 Off |                  Off |
| N/A   38C    P0    92W / 225W |     81MiB /  5062MiB |     68%   E. Process |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      6812      C   gmx                                          70MiB |
|    1      6812      C   gmx                                          70MiB |
+-----------------------------------------------------------------------------+
```

**Figure 17: Using nvidia-smi on RUDENS.**

### Best practices for the jobs with intensive I/O

The user's work directory (`/home/<username>`) and the corresponding files are located on a NAS array. A benefit is that the directory is shared among all nodes. It is therefore is easy to use. A disadvantage is the insufficient speed for many parallel I/O operations (working with a large number of small files, analysis of sequencing data, etc.).

The directory `/scratch` allows to use the fast local SSD disk on a compute node. It is intended to be used only during job execution, provides space for 200 GB – 960 GB (depending on the node), and requires manual copying of files to and from it. Users are encouraged to use the `/scratch` directory for data intensive jobs.

The following gives an example of how to use the `/scratch` directory. The commands can be executed directly on a compute node by using a batch script, or in an interactive session:

```
$ SCRATCH=/scratch/$PBS_JOBID
$ mkdir -m 700 $SCRATCH

# copy data to scratch
$ cp $HOME/input_data $SCRATCH

# run program
$ ./my_prog -input $SCRATCH/input_data -ouput $SCRATCH/output_data

# copy data back to home
$ cp -r $SCRATCH/output_data $HOME/
$ rm -rf $SCRATCH
```

It should be noted that not all nodes are equipped with local SSD disks. It is therefore recommended to specify an appropriate feature parameter when submitting a job, e.g., if a larger (960 GB) disk space is required the following requirement can be specified:

```
# PBS -l feature=largescratch
```

Multiple I/O-intensive jobs or parallel processes on the same compute node may overload network storage and reduce job efficiency.

To perform only a single user task on each compute node, the following job requirements need to be added:

```
# PBS -W x=naccesspolicy:UNIQUEUSER
```

The user should choose the optimal number of parallel processes (threads). It is important to mention that a higher number not necessarily means an increase in speed.

### *5.3.5 Using GPUs and CUDA on the cluster*

This section describes best practices for using GPUs and CUDA framework on the HPC cluster.

**Requesting GPUs for a job**

GPUs can be requested similarly to other cluster resources:

```
$ qsub -l nodes=1:ppn=1:gpus=1:shared
```

Additionally, it is possible to specify the compute mode for a GPU, i.e.,

- `shared`: the GPU is available for multiple processes
- `exclusive_process`: only one compute process is allowed to run on the GPU

To request a specific GPU model, additionally the feature parameter `-l feature=v100` needs to be specified.

The GPUs available on the cluster RUDENS are listed in the following Table 13. More details can be found in Sec. 5.1.1.

| GPU model | Arch | CUDA | FP64 power | Tensor power | Memory | feature (qsub) |
|---|---|---|---|---|---|---|
| Tesla K40 | Kepler | 3.5 | 1.3 GFLOPs | N/A | 12 GB | k40 |
| Tesla V100 | Volta | 7.0 | 7.5 GFLOPs | 112 TFLOPs | 16 GB | v100 |

**Table 13: GPUs available on RUDENS.**

### Development of GPU code using the CUDA Toolkit

The Compute Unified Device Architecture (CUDA) is a parallel computing platform and Application Programming Interface (API) model created by NVIDIA. It allows software developers to use a CUDA-enabled GPU for general purpose processing, an approach known as General Purpose GPU (GPGPU) computing. More information on the availability of GPUs in RUDENS can be found in the previous section.

To use CUDA, the user needs to load a version of the CUDA library (and compiler). Several versions of the CUDA library are available. They can be listed with the `module avail cuda` command. Different versions of CUDA are activated via the command:

```
$ module load cuda/cuda-<version>
```

To load the latest version, the command

```
$ module load cuda
```

can be used.

The following CUDA tools/libraries are installed on the RUDENS cluster:

- C++ extensions for CUDA programming
- CuDNN library for neural networks
- CuBLAS
- NCCL

Corresponding job examples are available at /opt/exp_soft/user_info/cuda.

The Linux-related graphical tools `emacs` and `ddd` can be used for programming and debugging:

```
$ module load cuda
$ emacs hello-world.cu
$ nvcc -g -G hello-world.cu –o hello-world.out
$ ddd –debugger cuda-gdb hello-world.out
```

Furthermore, NVIDIA `Nsight`[48], a development environment based on Eclipse and development by NVIDIA, can be used:

```
$ module load cuda
$ nsight
```

It should be noted that to be able to use GUI tools, `X11` forwarding must be enabled.

To achieve the best possible performance whilst being portable, GPU code should be generated for the architecture(s) it will be executed upon.

This is controlled by specifying the `-gencode` arguments to `nvcc` which, unlike the `-arch` and `-code` arguments, allows for 'fatbinary' executables that are optimized for multiple device architectures.

Each `-gencode` argument requires two values, the virtual architecture and the real architecture, for the use in `nvcc`'s two-stage compilation. That is, `-gencode=arch=compute_60, code=sm_60` specifies a virtual architecture of `compute_60` and real architecture `sm_60`.

The minimum specified virtual architecture must be less than or equal to the GPU's compute capability used to execute the code.

To build a CUDA application which targets any GPU on the HPC cluster RUDENS, the following `-gencode` arguments (for CUDA 8.0) should be used:

```
$ nvcc filename.cu \
  -gencode=arch=compute_35,code=sm_35 \
  -gencode=arch=compute_60,code=sm_60 \
```

### *5.3.6 Running machine learning applications*

There is a range of tools on the cluster to support the development of Machine Learning (ML) applications. Users can install various ML frameworks through the `Anaconda` environment. They can also use `Singularity` to load pre-built containers and use the CUDA libraries as described in the previous section.

#### Conda / Anaconda environment

The package and environment management system allows users to create isolated working environments, install various programming tools, including the recent `Python` versions and ML modules.

The working environment is prepared by first loading the module:

```
$ module load conda
```

[48] NVIDIA NSight https://developer.nvidia.com/nsight-visual-studio-edition

On the cluster, then the following can be executed:

```
$ conda create -n my_conda_enev python=3.9
$ conda info --envs
$ source activate my_conda_env
$ conda install pytorch torchvision -c pytorch
```

### Running containers with Singularity

Singularity can be accessed by firs loading the required module:

```
$ module load singularity
$ module list
```

The main online container registries and commands to load them are listed in Table 14.

| | | |
|---|---|---|
| Singularity Library | https://cloud.sylabs.io/library | `$ singularity pull library://` |
| Docker Hub | https://cloud.sylabs.io/library | `$ singularity pull library://` |
| Singularity Hub | https://cloud.sylabs.io/library | `$ singularity pull library://` |
| NVIDIA GPU Cloud | https://ngc.nvidia.com | `$ singularity pull docker://nvcr.io/` |

**Table 14: Containers registries and commands.**

The following is an example to load a container:

```
$ singularity pull docker://gcc:5.3.0
$ singularity pull library://godlovedc/demo/lolcow
```

To execute the containers on the HPC cluster, an interactive session can be started in the fast queue:

```
$ qsub -I -X -q fast
```

Alternatively, an interactive session can be run on a single node with Tesla K40m GPU. In this example the wall clock time is set to 1h and the `fast` queue is used:

```
$ qsub -I -l nodes=1:gpus=1,feature=K40,walltime=01:00:00 -q fast
```

The run command executes a defined set of commands from a definition file's run script. It is only available when using an image that was built from a definition file that specified a run script:

```
# Running container's runscript from library
$ singularity run library://godlovedc/demo/lolcow

# Execute runscript of loaded container
$ singularity run ./lolcow_latest.sif

# Change to working directory
$ cd $PBS_O_WORKDIR
```

The `exec` command executes command from the container:

```
$ singularity exec ./lolcow_latest.sif fortune
$ singularity exec library://godlovedc/demo/lolcow fortune

# Execute runscript of loaded container
$ singularity run ./lolcow_latest.sif

# Change to working directory
$ cd $PBS_O_WORKDIR

# Load Singularity module
$ module load singularity
```

The `shell` command allows to spawn a new shell within the container and interact with it as though it were a small virtual machine

```
$ singularity shell library://godlovedc/demo/lolcow
$ singularity shell container.sif

# Execute runscript of loaded container
$ singularity run ./lolcow_latest.sif

# Change to working directory
$ cd $PBS_O_WORKDIR

# Load Singularity module
$ module load singularity
```

User-defined bind paths can be specified via:

```
$ singularity shell --bind /scratch container.sif
```

If the host system has an NVIDIA GPU card and a driver installed, the user can leverage the card with the `--nv` option:

```
$ singularity run --nv container.sif
```

The following is an example of how to use the TensorFlow Deep Learning Framework[49] from the Docker container library[50]. The full output has been omitted from the following example for brevity:

[49] TensorFlow https://www.tensorflow.org
[50] Docker https://www.docker.com

```
$ singularity pull docker://tensorflow/tensorflow:2.3.1-gpu
$ qsub -I -l nodes=1:ppn=4:gpus=1
sub: waiting for job 1004934.RUDENS to start
qsub: job 1004934.RUDENS ready
$ module load singularity
$ cd $PBS_O_WORKDIR
@wn58 tensorflow]$ singularity exec --nv tensorflow_2.3.1-gpu.sif \
 python -c 'import tensorflow as tf; print(tf.__version__)'
2.3.1
@wn58 tensorflow]$ singularity exec --nv tensorflow_latest-gpu.sif \
python -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
…
Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 10610 MB
memory) -> physical GPU (device: 0, name: Tesla K40m, pci bus id: 0000:04:00.0, compute
capability: 3.5)
tf.Tensor(654.7006, shape=(), dtype=float32)
@wn58 tensorflow]$ singularity shell --nv tensorflow_2.3.1-gpu.sif
Singularity tensorflow_2.3.1-gpu.sif:~> python /usr/local/lib/python3.6/dist-
packages/tensorflow/python/debug/examples/debug_mnist.py
Accuracy at step 0: 0.216
Accuracy at step 1: 0.098
Accuracy at step 2: 0.098
```

For further reading, the reader is referred to the various documentations[51] [52] [53].

### *5.3.7 Usage accounting*

Cluster resources (CPU/GPU hours) consumed by the jobs are recorded by the `Moab` Accounting Manager[54]. Only the job execution time is taken into account. The time a user spends preparing the job or the queue waiting time is not billed. The cost of each job is automatically calculated according to the usage agreement of the HPC services. The account balance can be displayed with the commands:

```
$ mam-balance
$ mam-list-funds
```

To get a detailed report of cluster resources consumed by each job, for example, for July 2018, the following command can be used:

```
$ mam-list-usagerecords -s 2018-07-01 -e 2018-08-01
```

This results in the output shown in Figure 18.

```
[ciko@master ~]$ mam-list-usagerecords -s 2018-07-01 -e 2018-08-01
User Instance Nodes Processors GPUs Duration Memory SubmitTime          StartTime           EndTime
---- -------- ----- ---------- ---- -------- ------ ------------------- ------------------- -------------------
ciko 349694       1          1           31        2018-07-01 07:39:18 2018-07-01 07:39:19 2018-07-01 07:39:50
ciko 349695       1          1            1        2018-07-01 07:39:27 2018-07-01 07:39:50 2018-07-01 07:39:51
ciko 353029       1          1          124        2018-07-23 12:10:47 2018-07-23 12:10:52 2018-07-23 12:12:56
```

**Figure 18: Accounting report on RUDENS.**

[51] Singularity documentation https://www.sylabs.io/docs/
[52] Singularity user guide https://sylabs.io/guides/3.2/user-guide/
[53] Singularity quick start https://sylabs.io/guides/3.2/user-guide/quick_start.html
[54] Moab Accounting Manager http://www.adaptivecomputing.com/products/capabilities/accounting/

# 6 CLAIX at RWTH Aachen University

The IT Center has been operating HPC systems for many years to support research and teaching at RWTH Aachen University. The current system is called CLAIX (Cluster Aix-la-Chapelle) and consists of three parts: the Tier-2 shares (national system) from the 2016 and 2018 procurement phases, and a Tier-3 share (local system). Only the first two are relevant for this project. The up-to-date technical documentation for the RWTH Compute Cluster services provided by the IT Center can be found online on RWTH's Compute Cluster website[55], The key parts are replicated in this section.

In the following, first, details of the hardware specifications are presented in Sec. 6.1. Subsequently, Sec. 6.2 and Sec. 6.3 explain how to access the system and how to use it.

## 6.1 Hardware specifications

The system consists of two parts installed in 2016 and 2018, which are subsequently described in Sec. 6.1.1 and Sec. 6.1.2.

### *6.1.1 Tier-2 system CLAIX-2016*

The system consists of just over 600 nodes equipped with 2x Intel Broadwell processors. Specialized node types with up to 144 computing cores and 1 TB main memory or integrated GPUs or NVRAM supplement the system for special tasks. All nodes and the parallel Lustre[56] file system with a capacity of 3PB are interconnected through an Intel Omni-Path network with 100 Gb/s. The overall system achieves a computing power of approx. 670 TFLOPs.

### *6.1.2 Tier-2 system CLAIX-2018 (with heterogeneous nodes)*

CLAIX-2018 consists of 1,032 computing nodes with 2x Intel Skylake processors, each with 24 cores and 192 GB RAM. Also, there are 48 computing nodes of identical architecture, each equipped with two NVIDIA Volta V100 GPUs (incl. NVLink) as accelerators. They are available for special applications, e.g., for tasks performing machine learning.

To work with the system interactively, CLAIX also has eight additional dialogue systems equipped with the same CPUs but with 384 GB more RAM. All the CLAIX 2018 computing nodes are connected to an Intel Omni-Path 100 Gb/s network. A high-performance Lustre-based storage system offers a file system capacity of 10 PB and a bandwidth of 150 GB/s (read and write). The parallel file system is available via the environment variable `$HPCWORK`, see Sec. 6.3.1.

## 6.2 Accessing the system

As a Tier-2 system in the HPC supply pyramid of the Gauß-Allianz (GA)[57] in Germany, researchers from all over Germany time on the system.

To use computing resources at RWTH, it is necessary to have a compute-time project. The application process is in line with the recommendations of the Gauß-Allianz[58] for the

[55] RWTH Compute Cluster https://www.itc.rwth-aachen.de/go/id/hisv
[56] Lustre https://www.lustre.org
[57] Gauß-Allianz https://gauss-allianz.de
[58] Gauß-Allianz recommendation https://gauss-allianz.de/files/pdf/GA-Empfehlungen_Beantragungs_Bewilligungsverfahren_201709.pdf

establishment of nationally coordinated application and approval procedures. The procedure is shown in Figure 19. See RWTH's access policy[59] for more information.


**Figure 19: Chart of the application and approval process for HPC resources**

The category of the applications to be submitted and the scope for the evaluation, concerning, for example, the number and status of reviewers involved, depend on the volume of the compute-time requested. The up-to-date overview of the supported project types is documented online[60].

There exist different ways of obtaining compute time on the HPC systems at RWTH Aachen University. The most prominent track is applying for computing time on the Jülich Aachen Research Alliance - HPC (JARA-HPC) partition, which currently provides 2.4 Mcore-h per year. This partition is intended for large-scale computing projects run by researchers from Aachen and Jülich. Another option is going over BUND, which accepts applications from researchers from other German research institutions.

External scientists from any German research institution can write an application, identified by an automatically generated unique application-id, and once reviewers have approved this application, a project-id is assigned, and an HPC project account is generated.

In the following, the application process is described in detail:

1. The applicant, the compute-time project's PI and/or his/her contact person, is initially identified by his/her unique email-address. Whenever the applicant accesses the JARDS online application system[61], the applicant's identity is first checked by an email-callback method.
2. After navigating through the link sent by email, the applicant can then (continue to) fill the online application forms and upload a project description.
3. After finalizing the application, JARDS sends the filled application forms to the applicant in a single PDF file.
4. The applicant has to read, check, and sign a printed version of this PDF file and send it to the IT Center's ServiceDesk[62].
5. Once the IT Center receives this paper version, it creates a case in the electronic ticket system, which sends an email with a ticket number in the format `[#yyyymmdd-nnnn]` as part of the subject line to the applicant. This ticket is used to keep track of the whole application procedure.

[59] RWTH access policy https://help.itc.rwth-aachen.de/service/rhr4fjjutttf/article/04d4da051b004a208c92c1ce216b116a
[60] RWTH HPC projects https://www.itc.rwth-aachen.de/cms/IT-Center/Forschung-Projekte/High-Performance-Computing/~cutxf/Projektbewirtschaftung/lidx/1/
[61] JARDS online application system https://jards.itc.rwth-aachen.de/jards/WEB/review/login.php
[62] IT-Service desk servicedesk@itc.rwth-aachen.de

6. The technical reviewer checks the application and contacts the applicant if there is any formal or technical question. The technical reviewing process typically takes about one week.
7. The IT Center's system administrators need to know the username of the applicant, which is usually in the format `ab123456`, before they can generate an HPC project account. Applicants who already have a username have to provide their id as one of the application form entries.
    - For external applicants, JARDS initiates an additional process to provide such a username and the corresponding HPC account.
    - JARDS sends an email to the applicant, including a one-time web link, which the applicant has to visit and accept the RWTH Compute Cluster's terms of use.
    - After acceptance of these terms of use, another email is sent to the applicant containing a coupon code and another one-time web link.
    - The applicant visits this web site, enters a coupon code to generate a unique username and adds his/her email address.
    - The applicant receives another email with a one-time web link to verify that this email address is connected to the corresponding Identity Management (IDM) system[63].
    - The applicant visits this one-time web page, enters the first and last name and receives a username and password.
    - Again, the applicant receives an email from IDM, including a one-time web link.
    - On this one-time web page, the applicant has to confirm that the JARDS-coupon code has to be connected to the username.
    - The HPC account, which corresponds to the applicant's IDM id, is then generated automatically.
    - Finally, the applicant's email address is added to the RWTH Compute Cluster's mailing list (rzcluster@list.rwth-aachen.de), which is used to distribute important information concerning the cluster's operation.
    - Using the IDM, the applicant still has to define the password for the HPC account**.**
    - The connection of the PI's / PC's unique username to the coupon code is essential for later adding additional external project members to the HPC project account.
8. Once all technical questions are resolved, the project-id is determined and the HPC project account is generated and connected to the applicants personal HPC account. An initial small (test) quota is set, and an email is sent to the applicant.
9. Work can then start using the computer cluster consuming the initial test quota.
10. Once the scientific reviewers approve the application, this quota will be increased to the amount that is determined by the reviewers. The scientific reviewing process roughly takes one month.

Alternatively, PIs can invite externals, also from other countries, as long as the German Federal Office for Economic Affairs and Export Control (BAFA)[64] has no objections, to participate in an approved compute-time project as project members. In this case externals will receive their username and their HPC user account as part of a related coupon process.

[63]Identity Management system https://idm.rwth-aachen.de/
[64]BAFA http://www.ausfuhrkontrolle.info/

- A PI/PC who is a RWTH member can use the RWTH IDM partner mechanism to register additional external project members. The PI/PC acts as a sponsor for the external partner and is responsible for this partner.
- Using the RWTH Partner Manager online portal[65], the PI / PC generates a coupon code which the external partner can use to get an RWTH TIM-Id.
- An external PI/PC does not have the privileges to act as a sponsor. In this case, the cluster-access web portal[66] can be used to invite and register additional external project members.

**Getting support**

The IT Center of RWTH Aachen University provides user advising services and technical and methodological support, and is responsible for the continuous development of the cluster. Furthermore, the IT Center is responsible for the procurement of systems to meet local, regional, and subject-specific national computing power requirements. Support requests can always be made via the IT-ServiceDesk, which serves as a single point of contact for users.

The IT Center assists with questions regarding its HPC offerings (account creation, login, use of the cluster/batch system, installation of software, etc.), including, in particular, support for performance analysis and optimization. Furthermore, the IT Center also offers a comprehensive program of courses and workshops, e.g., PPCES[67], aiXcelerate[68]. The course program is supplemented by an extensive documentation in the HPC.Wiki[69].

## 6.3 Using the system

Following the "one-cluster concept", the operational strategy makes all the cluster resources available to the users via an interface. Different expansion stages, innovative architectures and data can be used by means of the same processes. The one-cluster concept has developed from these requirements. It aims to operate all components within a large cluster and offers the following advantages:

- The same interfaces are available to users within the cluster concerning identity management, dialogue systems, workload management system, operating system, software stack, and file systems. The necessary knowledge of key components remains limited for users, the focus on one interface facilitates communication, and documentation is easier to maintain.
- By using an own solution for cluster management, operational processes scale optimally and can be adapted to different scenarios. For example, linking the various cluster management tools allows changes to be made throughout the entire cluster based on monitoring data, as these are based on the same technical foundation.
- This has made it possible for years to operate the HPC system without fixed maintenance windows, resulting in very high availability of the system with very few interruptions in operation. Such interruptions are only required in exceptional cases, such as maintenance work on the file systems or major modifications such as a change

[65] RWTH Partner Manager portal http://www.rwth-aachen.de/partner-manager
[66] Cluster access portal https://www.jards.itc.rwth-aachen.de/cluster-access/
[67] PPCES http://itc.rwth-aachen.de/ppces
[68] Aixcelerate http://itc.rwth-aachen.de/aixcelerate
[69] HPC.Wiki http://hpc-wiki.info/

in the operating system. For small maintenance tasks such as the installation of new kernel versions, the batch system is used, which means no restrictions for the user.
- Clustering, in this way, leads to highly scalable operational processes and enables new and innovative functions to be available immediately on all suitable architectures and expansion levels. Also, this makes it possible to set up a large number of new systems in a very short time and to integrate them into the cluster, for example, when expanding with a new expansion stage.
- User-side differentiations, e.g., between processor architectures or server types, are made to take the technological-scientific assessment as part of the approval process into account; on the operational side, such differentiation is kept as low as possible.

The dialogue systems constitute the user interface to the HPC systems. These can be used to prepare, commission, control, and evaluate calculation requests, and to use development and analysis applications. Using special copy nodes with a broadband connection to the university and institutional research network, large amounts of data can be transferred to and from the HPC systems. The large groups of backend systems (CLAIX-2016, CLAIX-2018, Tier-3, Innovative Architectures (GPU, KNL), Integrative Hosting) are made available through the Workload Management System in Sec. 6.3.3. They are not directly accessible. The file systems can be accessed from the entire cluster and can be addressed by the users as `$HOME`, `$WORK`, and `$HPCWORK`. Large parts of the individual backend groups are interconnected via high-performance and redundant Omni-Path networks.

### *6.3.1 Available file systems and data transfer*

For the storage of data, the users of the HPC systems can choose between different file systems, depending on the intended usage scenarios. The file systems vary in certain characteristics, including performance metrics, available space, and data protection concepts. The following file systems can be used:

- $`HOME` is an NFS-based file system that provides users with 150 GB of standard disk space to store the most important data such as source codes and configuration files. The use of snapshot mechanisms and safe data storage in the backup system of RWTH Aachen University guarantees a very high level of data security. This is also reflected in the 100% availability of the file system since 2016.

- `$WORK` is also an NFS file system. However, its technical structure is intended for storing larger files. This includes, e.g., the results of performed calculation jobs. With 250GB, more storage space is available to users of this file system. However, this data is not backed up, so that it should be reproducible. Accidentally deleted files, however, can be recovered based on snapshots.

- `$HPCWORK` consists of two file systems based on the parallel high-performance file system Lustre. This file system is characterized by high read and write rates. With a standard storage capacity of 1TB, the space available is significantly higher than on the other file systems. Due to the amount of data, central data backup is, however, not possible.

### *6.3.2 Available software*

The software stack of the cluster is managed by the IT Center and is partly developed by the IT Center itself. This approach has long been followed and offers a number of advantages:

- The independence from specific manufacturers ensures flexibility and allows quick adaptations to the frequently changing requirements in research and teaching, e.g., for the integration of innovative architectures.
- Saving of license and maintenance fees for software, e.g., for operating systems.
- Access to all layers of the software stack allows effective and efficient error and performance analyses as well as comprehensive changes in response to analysis results.
- Consistent pursuit and implementation of an open source strategy.

The operating system used is CentOS, an open Red Hat-based Linux variant.

About 100 different Independent Software Vendor (ISV) and open source software packages are made available to users in various subject-specific categories. The IT Center is responsible for providing and maintaining such software if there is sufficient demand and operates the necessary license servers, if necessary. In particular, tools for using the cluster (e.g. graphical interfaces), parallelization (various MPI implementations), programming (compilers, libraries) and application analyses (debuggers, performance analysis, and visualization) are centrally provided to the users.

CLAIX offers the typical machine learning frameworks, including GPU support, see Sec 6.3.4. This includes `TensorFlow`, `PyTorch`[70], and `Horovod`[71], for example.
The up-to-date list of available software and license restriction[72] can be found in the documentation portal[73].

To handle the different requirements of each piece of software, different compilers, MPI implementations, tools, and so forth, the system provides a module system, which allows the user to easily load the needed software and/or switch between its different versions. All necessary search paths and environment variables are set by loading a module. The modules package is available for the `ksh`, `zsh`, and `tcsh` shells. Further information can be found documentation portal[74].

### *6.3.3 Batch system*

The `SLURM` workload management system, cf. Sec. 2.3.4, is used to manage the compute jobs on the back-end systems. Extensive documentation on how to use the batch system has been put into the HPC.Wiki's section on the Batch Schedule[75]. Again, documentation specific to the IT Center can be found in the documentation portal[76].

[70] PyTorch https://pytorch.org
[71] Horovod https://github.com/horovod/horovod
[72] Restriction list https://help.itc.rwth-aachen.de/service/rhr4fjjutttf/article/022b08d321fb45609bedb391ad46ef42
[73] RWTH documentation portal http://help.itc.rwth-aachen.de/
[74] RWTH module documentation https://help.itc.rwth-aachen.de/service/rhr4fjjutttf/article/417f822b8a7849eb8c9c2753045ad67f
[75] Batch scheduling at RTWH https://hpc-wiki.info/hpc/Batch-Scheduler
[76] Batch scheduling documentation portal https://help.itc.rwth-aachen.de/service/rhr4fjjutttf/article/13ace46cfbb84e92a64c1361e0e4c104

The following example submits a hybrid parallel program to CLAIX:

```
#!/bin/zsh
### Job name
#SBATCH --job-name=HelloHybrid
### 2 compute nodes
#SBATCH --nodes=2
### 4 MPI ranks
#SBATCH --ntasks=4
### 2 MPI ranks per node
#SBATCH -ntasks-per-node=2
### 3 tasks per MPI rank
#SBATCH --cpus-per-task=3
### the number of OpenMP threads
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
### Change to working directory
cd /home/usr/workingdirectory
### Run your parallel application
srun hello.exe
```

On CLAIX, it is not necessary to choose a partition for the job, as this is done by the job modifier script depending on the project and resources the user is asking for. Every project has a default partition and (eventually) additional partitions, it is allowed to submit to. However, different partitions are supported, which are listed in Table 15.

| partition | #nodes | #cores/node | #mem/node [GB] | remarks |
|---|---|---|---|---|
| c18m | 1,240 | 48 | 192 | default partition for default project |
| c18g2 | 54 | 48 | 192 | 2 V100 GPUs, request of Volta GPU needed to submit to this partition |
| c16m3 | 608 | 24 | 128 | project needed to be able to submit to |
| c16s4 | 8 | 144 | 1024 | project needed to be able to submit to |
| c16g6 | 8 | 24 | 128 | 2 P100 GPUs, request of Pascal GPU needed to submit to this partition |

**Table 15: Partitions on CLAIX.**

### *6.3.4 GPU Computing*

CLAIX 2018 provides 48 nodes equipped with two NVIDIA V100 GPUs each. Furthermore, there are special nodes for interactive use available[77]. Please note that these nodes are for

[77] CLAIX special nodes https://help.itc.rwth-aachen.de/service/rhr4fjjutttf/article/a1beccd9e9dc4044a740ed248f478839

short (less than ten minutes) tests only. Only batch jobs should be used for real applications. To this end, the `--gres=gpu:<no. GPUs>` parameters need to be added to the `SLURM` batch file. A simple example to run `deviceQuery` (from NVIDIA SDK) on one device is:

```
#!/usr/local_rwth/bin/zsh
#SBATCH -J single-gpu
#SBATCH -o single-gpu.%J.log
#SBATCH --gres=gpu:1
module load cuda
#print some debug informations...
echo; export; echo;  nvidia-smi; echo
$CUDA_ROOT/extras/demo_suite/deviceQuery -noprompt
```

To run an MPI application on multiple GPU nodes, the number of MPI processes needs to match the number of GPUs requested:

```
#!/usr/local_rwth/bin/zsh
#SBATCH -J multi-gpu
#SBATCH -o multi-gpu.%J.log
#SBATCH --ntasks=4
#SBATCH --ntasks-per-node=2
#SBATCH --gres=gpu:2
module load cuda
#print some debug informations...
echo; export; echo; nvidia-smi; echo
MPIEXEC $FLAGS_MPI_BATCH ./cuda-mpi
```

# 7 GARPUR and LUMI at University of Iceland

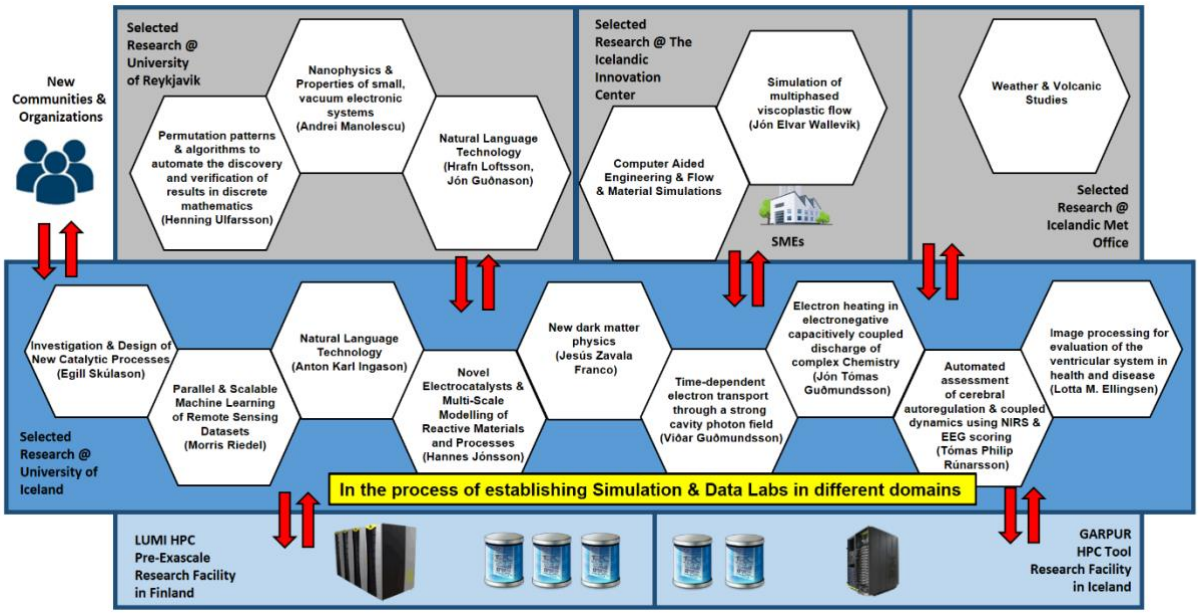



**Figure 20: The national competence center for AI & HPC of Iceland is primarly focused on domain-specific science activities that take advantage of the GARPUR system and the LUMI supercomputer in the future.**

The requirements for HPC systems by its users at the University of Iceland (UOI) in particular but Iceland, in general, are quite diverse. Independent research groups are driven by specific PIs and use HPC for various scientific applications on different levels of scale. In the EuroCC project[78], the national competence center (NCC) Iceland establishes Simulation & Data Labs (SimDataLabs) with those PIs. With these labs it is intended to have a more structured HPC community in Iceland that applies more consistently for HPC funding. This SimDataLabs model implements and adapts the research and support structure of the Simulation and Data Labs[79] running successfully at JSC for more than 15 years. Figure 20 summarizes the different key areas of science with the PIs and the underlying HPC infrastructure setup that consists of the GARPUR[80] research facility in Iceland and the LUMI[81] HPC pre-exascale system in Finland.

Based on the various research groups' requirements, the Icelandic infrastructure roadmap for HPC implements a step-wise strategy. While beginners of HPC use the JÖTUNN[82] cluster in university courses or tutorials, HPC end-users typically use the GARPUR system for applications with moderate requirements for scaling. Only after end-users have successfully shown that they can scale up their applications on GARPUR, they will be granted access to the LUMI system. The pan-European flagship supercomputer LUMI is financed by ten different countries. Iceland is a rather small partner in the LUMI consortium. Consequently, only a few users will run applications on LUMI. This low number of users also reflects the current HPC user communities' scaling demands in Iceland. Some PIs have also access to PRACE HPC resources via double affiliations or long-term collaborations.

In the following, the GARPUR and LUMI systems' hardware specifications are provided in Sec. 7.1 Subsequently, Sec. 7.2 discusses how to access the HPC system GARPUR before

[78] EuroCC https://www.eurocc-project.eu/
[79] Simulation and Data Labs https://www.fz-juelich.de/ias/jsc/EN/Expertise/SimLab/simlab_node.html
[80] GARPUR https://ihpc.is/garpur/
[81] LUMI https://www.lumi-supercomputer.eu/
[82] JÖTUNN https://ihpc.is/jotunn/

Sec. 7.3 explains how to use HPC systems efficiently. For continuously updated documentation, the reader is referred to the GARPUR[83] and LUMI[84] supercomputer documentations.

## 7.1 Hardware specifications

For the CoE RAISE, the production system GARPUR and the upcoming LUMI system are relevant for developments. Their hardware specifications are given in the following Sec. 7.1.1 and Sec. 7.1.2.

### *7.1.1 GARPUR*

GARPUR is a joint project between the UOI and the University of Reykjavík with funding from the Icelandic Centre for Research (Rannís)[85]. Research topics of computations performed on GARPUR range from transport in quantum wires to ice sheet modelling of glaciers. GARPUR was opened for users in late April 2016[86]. In late 2017, GARPUR received an upgrade which more than doubled the performance of the cluster. Recently, another major upgrade is foreseen for 2020. The update will be funded by Rannís via the Icelandic Research High Performance Computing (IRHPC) grant that will change the system significantly. Hence, at the time of writing, GARPUR consists of the components shown in Table 16.

| **Queue** | **`normal`** |
|---|---|
| Nodes | 36 |
| CPU | 2x Intel Xeon E5-2680v3 (2.5GHz, 12 cores) |
| Memory | 128 GB (8x16 GB DDR4 2133MHz) |
| Interconnect | 56 Gb/s (FDR) Infiniband network |
| Disk | 1 TB /scratch |
| **Queue** | **`himem`** |
| Nodes | 8 |
| CPU | 2x Intel Xeon E5-2698v3 (2.3GHz, 16 cores) |
| Memory | 256 GB (16x16 GB DDR4 2133MHz) |
| Interconnect | 10 Gb/s ethernet network |
| Disk | 1 TB /scratch |
| **Queue** | **`himem-bigdisk`** |
| Nodes | 3 |
| CPU | 2x Intel Xeon E5-2698v3 (2.3GHz, 16 cores) |
| Memory | 256 GB (16x16 GB DDR4 2133MHz) |
| Interconnect | 10 Gb/s ethernet network |
| Disk | 4 TB /scratch |
| | |

[83] GARPUR docuymentation https://ihpc.is/garpur/
[84] LUMI documentation https://www.lumi-supercomputer.eu/lumi_supercomputer/
[85] Rannís https://en.rannis.is
[86] GARPUR opening https://www.hi.is/frettir/ny_ofurtolva_hja_reiknistofnun_haskola_islands

| **Queue** | **`Omnip`** |
|---|---|
| Nodes | 46 |
| CPU | 2x Intel Xeon Gold 6130 CPU (2.10GHz, 16 cores) |
| Memory | 192 GB DDR4 |
| Interconnect | 50Gb/s Omni-Path network |
| Disk | 140 GB /scratch |
| **Queue** | **`gpu`** |
| Nodes | 3 |
| CPU | 2x Intel Xeon E5-2648L (1.8GHz, 8 cores) |
| Memory | 64 GB (1x) & 80 GB (2x) |
| GPU | 2x Tesla M2090 |
| Disk | no local disk only NFS mounts |
| **Queue** | **vgpu** |
| Nodes | 2 |
| CPU | 2x Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz |
| Memory | 128 GB (8x 16 GB DIMM synchronous 2666 MHz (0.4 ns)) |
| GPU | Tesla V100 |
| Disk | 2 TB /scratch |

**Table 16: Hardware specification of the HPC system GARPUR.**

### *7.1.2 LUMI*

LUMI[87] is not yet installed but will become available during the runtime of RAISE. At the time of installation, LUMI will be one of the world's fastest computer systems, having theoretical computing power of more than 550 PFLOPs. LUMI's performance will be more than tenfold compared to one of Europe's fastest supercomputer today (Piz Daint[88], Switzerland). LUMI will also be one of the world's leading platforms for AI that is relevant to RAISE.

As a consequence of being not yet installed, the hardware specification is less detailed but will be made available to HPC users in RAISE in the course of the project once access is possible. As LUMI is part of the EuroHPC Joint Undertaking (JU) strategic endeavor, the access will partly also be provided via the JU.

LUMI's computing power will be equivalent to the combined performance of 1.5 million of the latest laptop computers. These would form an over 23km high tower. LUMI will have ample storage of 117 PB and an impressive aggregated I/O bandwidth of 2 TB/s. The supercomputer achieves its high performance with many nodes with accelerators (i.e., GPUs). The system is also complemented by a CPU-only partition, Infrastructure as a Service (IaaS) cloud services, and a large object storage solution.

LUMI will be using 100% hydro-powered energy. Up to 200MWs are available. The waste heat of LUMI will produce 20% of the district heat of the area. Reducing $CO_2$ emissions is a globally critical target, to which the location of the EuroHPC machines have a considerable impact, as supercomputers consume plenty of electricity. LUMI will be hosted in the Finland IT center for

[87] LUMI https://eurohpc-ju.europa.eu/news/lumi-new-eurohpc-world-class-supercomputer-finland
[88] Piz Daint https://www.cscs.ch/computers/piz-daint/

science (CSC) data center in Kajaani that has an abundant supply of low-price and environmentally-friendly, renewable power. The location's benefits also include warm water cooling, which enables waste heat to be utilized in the district heating network of Kajaani, further reducing costs, and $CO_2$ footprint. The CSC Kajaani data center is highly scalable w.r.t. large hardware installations and extensions or other potential infrastructure, making it ideal for building a sustainable data center ecosystem. The reliable and fast data communications networks of the data center are also designed for HPC. Besides, the area has a unique focus on data analytics thanks to the Kajaani University of Applied Sciences and numerous ICT companies.

The LUMI system will be a Cray EX supercomputer supplied by Hewlett Packard Enterprise (HPE). A summary of the current hardware specifications is provided in Table 17.

| | |
|---|---|
| **Sustained performance:** | 375 PFLOPs |
| **Peak performance:** | 552 PFLOPs |
| **Compute partitions:** | GPU partition (LUMI-G), x86 CPU-partition (LUMI-C), data analytics partition (LUMI-D), container cloud partition (LUMI-K) |
| **CPU:** | The LUMI-C partition will feature 64-core next-generation AMD EPYC™ CPUs. |
| **GPU:** | LUMI-G based on the future generation AMD Instinct™ GPU |
| **Storage capacity:** | LUMI's storage system will consist of three components. First, there will be a 7PB partition of ultra-fast flash storage, combined with a more traditional 80PB capacity storage, based on the Lustre parallel filesystem, as well as a data management service, based on Ceph and being 30PB in volume. In total, LUMI will have a storage of 117PB and a maximum I/O bandwidth of 2TB/s. |
| **Applications:** | AI, especially deep learning, and traditional large-scale simulations combined with massive scale data analytics in solving a single research problem. |
| **Other details:** | LUMI takes over 150$m^2$ of space, which is about the size of a tennis court. The weight of the system is nearly 150,000kg (150 metric tons) |

**Table 17: Hardware specification of the HPC system LUMI (at the time of writing).**

## 7.2 Accessing the systems

To use computing resources such as GARPUR, it is not necessary to have a compute-time project. Compute-time allocations are provided to specific PIs without the need to write a proposal for a large grant. More resource provisioning activities with compute-time allocations including peer-review processes are planned in 2020 under the umbrella of the EuroCC project. In the following, a short overview is given on how to apply for a GARPUR account. As

LUMI is not installed yet, there is no documentation available on this matter. The corresponding information will be provided to the RAISE members once the LUMI system is installed.

For the GAPUR system, PIs and the research groups' members receive an account via the Icelandic HPC support team[89] as follows, (a snapshot of the portal is shown in Figure 21).

All staff and students of UOI can use the service portal[90] for an account request and general HPC support by including “HPC” in the subject. There, the status of the request can also be monitored. Users outside UOI need to send a special access request to help@hi.is using “HPC” in the subject.

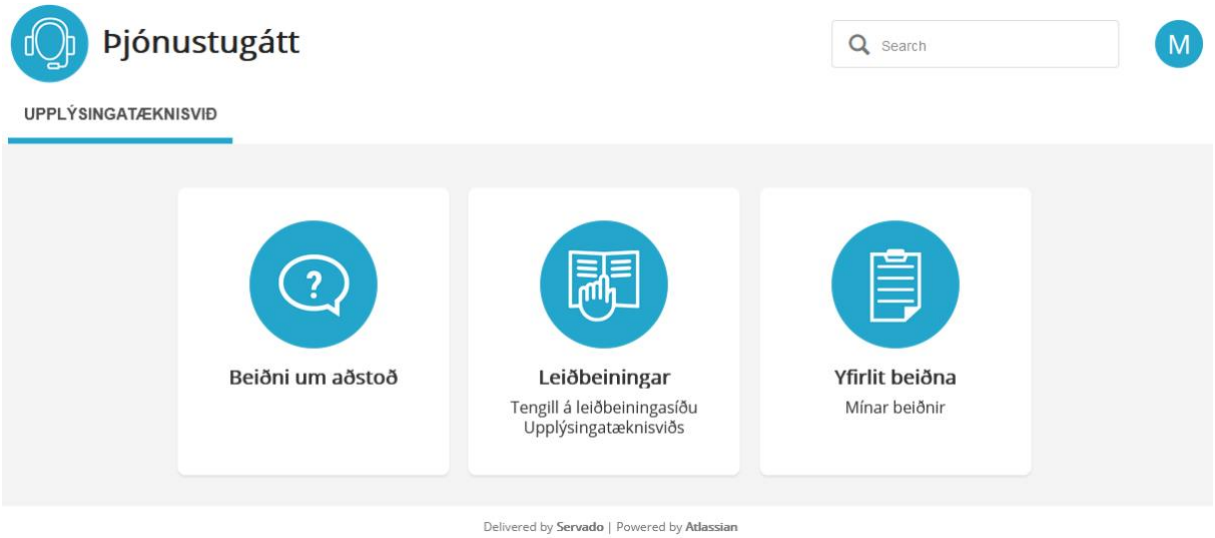



**Figure 21**: **Support portal of the University of Iceland is also used fort he HPC support requests, including the demand for receiving an account on GARPUR.**

## 7.3 Using the GARPUR system

The GARPUR HPC system is available via `ssh` using a username and password or by uploading puplic `ssh` keys and authenticating against the system with the corresponding private keys. The first step in connecting to GARPUR is thus setting up an `ssh` client. Most Linux system have such a client installed from the get-go, while Windows users are recommended to use, e.g., the free Windows `ssh` client MobaXterm[91]. All computations on the HPC systems are performed through a command-line interface. Users should always run their calculations on the compute nodes, not on the login nodes. This is done by using the `SLURM`[92] queuing system, cf. Sec. 2.3.4. If users are unfamiliar with these tools, it is recommended to first take a look at the YouTube[93] teaching material of the HPC course of UOI.

To run a job on the system, a job script needs to be created. A job script is a regular bash shell script with some directives, specifying the number of CPUs, memory etc. Then, this will be interpreted by the batch system on submission.

Below is an entire job sample script:

[89] Icelandic HPC support team https://ihpc.is/support/
[90] Service portal https://jira.hi.is/plugins/servlet/gateway?id=1#!/dashboard
[91] MobaXterm https://mobaxterm.mobatek.net/
[92] SLURM https://slurm.schedmd.com/overview.html
[93] YouTube UOI HPC course https://www.youtube.com/channel/UCWC4VKHmL4NZgFfKoHtANKg

```
#!/bin/bash
#SBATCH -n 1
#SBATCH -J ExampleJobName

#print hostname on which the job is running
echo $HOSTNAME
```

Once the job script is prepared, the `sbatch` command can be used as follows:

```
$ sbatch <your job script filename>
```

To check the status of the job the user can run:

```
$ squeue -u <username>
```

Other options of note for job scripts are

- `-N nodes=2 –ntasks-per-node=4` (requests 2 nodes with 4 cores each for the job)
- `-p himem` (requests to use the high memory nodes or GPU nodes)

If a job needs to be run on the compute nodes interactively, the following commands can be used:

```
$ salloc -N 1
$ ssh $SLURM_NODELIST
```

To leave this interactive session, the `exit` command can be used.

For more detailed job scripts, the reader is referred to the explanations of the computational molecular chemistry group[94].

Finally, GARPUR is actively monitored by the Ganglia system[95], which shows the current usage and load of the system.

[94] Comp. molec. chem. group https://sites.google.com/site/ragnarbjornsson/garpur-slurm
[95] Ganglia system http://garpur-main.ihpc.hi.is/ganglia/

# List of Acronyms and Abbreviations

| | |
|---|---|
| API | Application Programming Interface |
| BN | Booster Node |
| BSC | Barcelona Supercomputing Center |
| CIDR | Classless Inter-Domain Routing |
| CLAIX | Cluster Aix-la-Chapelle |
| CN | Cluster Node |
| CoE RAISE | European Center of Excellence in Exascale Computing “Research on AI- and Simulation-Based Engineering at Exascale” |
| CPU | Central Processing Unit |
| CSC | Finland IT Center for Science |
| CTE | Cluster de Tecnologies Emergents |
| CUDA | Compute Unified Device Architecture |
| DDR | Double Data Rate |
| DIMM | Dual Inline Memory Module |
| DRAM | Dynamic Random-Access Memory |
| EDR | Endpoint Detection and Response |
| FDR | Fourteen Data Rate |
| EFLOP | Exaflops |
| FZJ | Forschungszentrum Jülich GmbH |
| GA | Gauß-Allianz |
| GCS | Gauss Centre for Supercomputing |
| GPA | General-purpose architecture |
| GPFS | General Parallel File System |
| GPGPU | General-purpose graphics processing unit |
| GPU | Graphics Processing Unit |
| GUI | Graphical user interface |
| HBM | High Bandwidth Memory |
| HDD | Hard Disk Drive |
| HPC | High-performance computing |
| HPE | Hewlett Packard Enterprise |
| IaaS | Infrastructure as a Service |
| ICT | Information and Communication Technology |
| IDM | Identity Management |
| IRHPC | Icelandic Research High Performance Computing |
| ISV | Independent Software Vendor |
| I/O | input/output |
| ISV | Independent Software Vendor |
| JARA | Jülich Aachen Research Alliance |
| JURECA | Jülich Research on Exascale Cluster Architectures |
| JSC | Jülich Supercomputing Centre |
| JU | Joint Undertaking |
| JUST | Jülich Storage Cluster |
| JUWELS | Jülich Wizard for European Leadership Science |
| KNL | Knight’s Landing |
| MCDRAM | Multichannel DRAM |
| Mcore-h | Million core-hours |
| MIC | Many-Integrated Core |

| | |
|---|---|
| ML | Machine Learning |
| MPI | Message Passing Interface |
| MSA | Modular supercomputing architecture |
| NAS | Network-Attached Storage |
| NCC | National Competence Center |
| NFS | Network File System |
| NIC | John von Neumann-Institute for Supercomputing |
| NUMA | Non-Uniform Memory Access |
| NVMe | Non-volatile memory |
| OpenMP | Open Multi-Processing |
| PA | Project administrator |
| PFLOPs | Petaflops per second |
| PI | Principal investigator |
| PGI | Portland Group |
| PRACE | Partnership for Advanced Computing in Europe |
| PMT | Project management team |
| RAISE | see CoE RAISE |
| QDR | Quad Data Rate |
| QoS | Quality of Service |
| RAID | Redundant Array of Independent Disk |
| RAM | Random-Access Memory |
| Rannís | Icelandic Centre for Research |
| RTU | Riga Technical University |
| RWTH | Rheinisch-Westfälische Technische Hochschule Aachen |
| SATA | Serial AT Attachment |
| SDL | Simulation and Data Laboratory |
| SSD | Solid-state disc |
| TFLOPs | Teraflops per second |
| TORQUE | Terascale Open-source Resource and QUEue Manager |
| TSM | Tivoli Storage Manager |
| UOI | University of Iceland |
| VSR | Vergabe von Supercomputerressourcen |
| XFS | Extended File System |