Best practice guidelines / tutorials prototype

Table of Contents

List of Figures

List of Tables

Executive Summary

Before they are integrated into production systems, new high-performance computing developments are traditionally incorporated into pilot systems for testing purposes. Working with such systems can be quite challenging for new and novice users, and for experts that are accommodated to a different working environment.

This best-practice guideline is provided to lower the barriers to using the pilot systems available to RAISE and exploit their full potential in software development. It clearly describes the process from initial account application and/or creation at the respective supercomputing sites in RAISE to using the systems under the constraints of achieving optimal performance in a minimum number of steps. The document aims at serving as a reference for all developers in the project and stakeholders working on porting codes to the latest architectures on the market. It contains all necessary system usage information and avoids the redundancy of gathering information individually. It thereby accelerates the developments in RAISE.

Systems covered by this document are the DEEP and JUAWEI systems installed at Forschungszentrum Jülich and the CTE-ARM and CTE-AMD systems located at Barcelona Supercomputing Center.

1 Introduction

The development of high-performance computing (HPC) systems with heterogeneous architectures, with each component targeting specific stages in workflows of complex simulations or data analysis, is considered a milestone in reaching efficiency in the computation for applications with mixed and challenging demands. Such systems break with the rather inefficient concept of using a single architecture for multiple purposes and come up with the efficient approach of using multiple architectures for multiple purpose. The latter enables to combine architectures especially suited for specific tasks to reach fast and accurate computations.

The prototype systems used in RAISE serve as a blueprint for the next-generation exascale supercomputers. Hence, they are of particular interest to RAISE experts to drive algorithmic and methodological technologies towards exascale and an eye on energy efficiency. In more detail, the offered systems enable to port simulation and high-performance data analysis (HPDA) tools that are already available through the use-cases and from external sources to cutting-edge architecture, and to test their performance. New opportunities in data handling and workflows can be exploited by utilizing the underlying hardware technologies' features. Together with the systems' modularity, this enables to tweak applications with mixed hardware requirements and expand workflows to simultaneous and in-situ execution. That is especially relevant for applications driven by artificial intelligence (AI) methods given by the use-cases of RAISE, which are expected to benefit from system heterogeneity and parallel workflows massively.

At Forschungszentrum Jülich (FZJ), the latest hardware technologies are integrated into various pilot systems with the DEEP-EST system serving as a testbed to implement novel pre-exascale Modular Supercomputing Architecture (MSA) technologies. Subsequently, Sec. 2 provides the guidelines to use the DEEP-EST system at FZJ. This is followed by the descriptions of the ARM-based HUAWEI system JUAWEI at FZJ in Sec. 3. The ARM- and AMD-based systems CTE-ARM and CTE-AMD located at the Barcelona Supercomputing Center (BSC) are described in Sec. 4 and Sec. 5. Initially, it was planned to also provide information on a prototype installed at the Riga Technical University (RTU) in this document that included a BeeGFS file system. The file system was recently integrated in the production system RUDENS and the corresponding best-practice guide was shifted to the document “Best practice guidelines/tutorials for MSA/heterogeneous systems” of RAISE, i.e., the reader is referred to this report for more details.

This document is primarily intended for the experts in RAISE that will use the described prototype systems to develop and test novel AI technologies and simulation and HPDA software related to the use-cases of the CoE. As it will be publicly available, it can also serve as a starting point for novice users and developers new to the cutting-edge hardware technologies integrated into the prototype systems. It aims to gather all necessary information for the usage of the prototype systems in RAISE in one place and hence serves as a reference in the project’s software developments.

As the prototype landscape in RAISE is subject to continuous changes, this document represents the initial version of a living document that will continuously be updated and provided publicly on the CoE’s website[1].

[1] CoE RAISE Website https://www.coe-raise.eu

2 DEEP-EST system at Forschungszentrum Jülich

In agreement with the DEEP (Dynamical Exascale Entry Platform) projects[2], the DEEP-EST (DEEP - Extreme Scale Technologies) system, located at the Jülich Supercomputing Centre (JSC), FZJ, will be available for testing and development purposes in RAISE. Even though the DEEP projects soon come to an end, JSC clarified that the DEEP prototype systems are also usable after the project. While the systems JUWELS (Jülich Wizard for European Leadership Science) and JURECA (Jülich Research on Exascale Cluster Architectures) at FZJ are production MSA systems (see "Best practice guidelines/tutorials for MSA/heterogeneous systems" document of RAISE), the DEEP-EST system is a prototype. It aims at driving the modularity and heterogeneity concept and targets applications with diverse computational requirements.

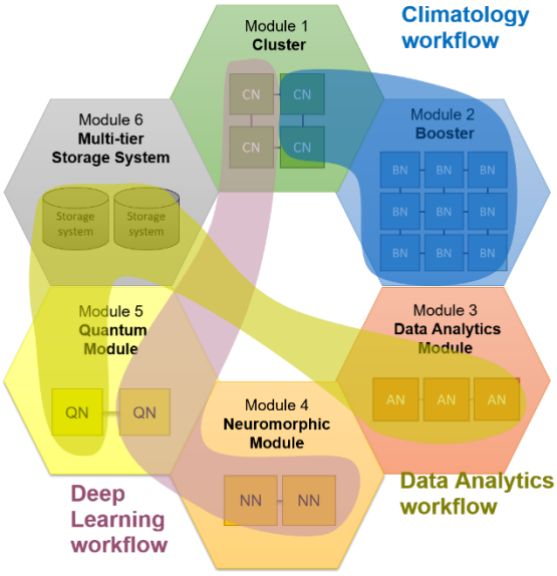

Figure 1: Using an MSA by applications with mixed computational requirements [1].

Despite general-purpose architectures (GPAs) are highly flexible and support many applications, they suffer from high energy consumption. Connecting additional modules such as a Booster suited for highly parallel execution or a module consisting of general-purpose graphics processing units (GPGPUs) to a GPA leads to the MSA approach. The continuously increasing complexity of computations with many different concurrently executed functionalities requires a strategy to minimize the energy consumption and the time to solution. This is achieved by the MSA, which enables to efficiently support such computations with heterogenous functionalities by placing specific workflow components on the best-possible hardware. Figure 1 shows an exemplary usage of an MSA by different kinds of applications.

In the following, a system overview of the DEEP-EST system will be given in Sec. 2.1. This is followed by a description of how to access the system in Sec. 2.2 and how to efficiently use it for development purposes in Sec. 2.3. A continuously updated Wiki is available through FZJ's corresponding DEEPTRAC website[3]. The user support is available by email (sc@fz-juelich.de).

2.1 DEEP-EST system hardware specifications

The DEEP-EST system consists of the following three major compute components:

- A **Cluster Module (CM)** with 50 compute nodes. Each node is equipped with two Intel Xeon 'Skylake' Gold 6146 with 12 cores (24 threads), clocked at 3.2GHz. All nodes are equipped with 192 GB RAM and a 400GB non-volatile memory express (NVMe) solid-state disk (SSD). The nodes are connected via an InfiniBand EDR (100 Gb/s) network. The following Figure 2 gives an overview of the hardware components integrated into a single node of the CM.

[2] DEEP projects https://www.deep-projects.eu
[3] DEEP-EST Wiki https://deeptrac.zam.kfa-juelich.de:8443/trac/wiki/Public/User_Guide

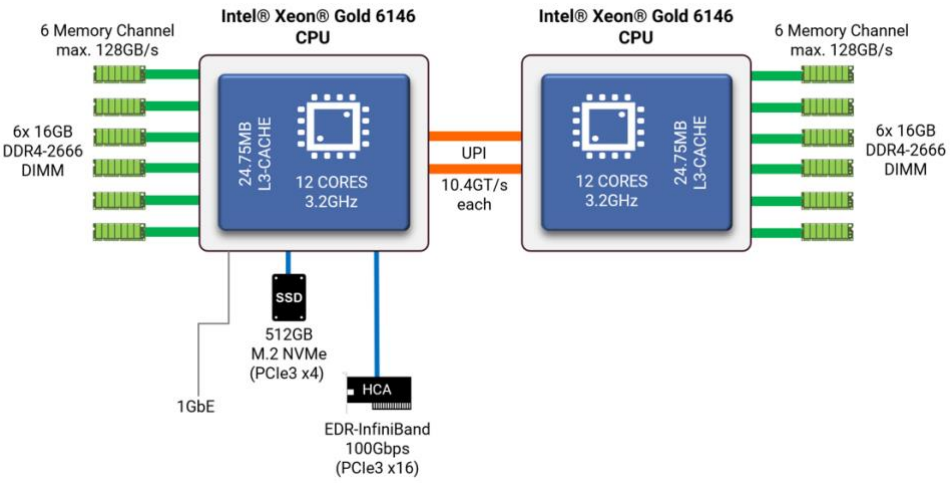

Figure 2: Hardware overview of a single node of the DEEP-EST Cluster module [1].

- An **Extreme-Scale Booster (ESB)** composed of 75 nodes. Each node is equipped with a single Intel Xeon 'Cascade Lake' Silver 4215 CPU, clocked at 2.50GHz, an NVIDIA V100 Tesla GPU with 32 GB high-bandwidth memory v2 (HBM2), 48 GB of RAM, and a 512 GB SSD. The nodes are connected via an EXTOLL (100 Gb/s) network. Figure 3 provides an overview of the ESB node configuration.

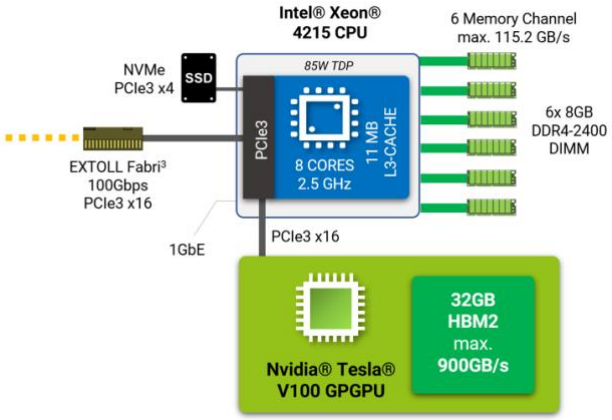

Figure 3: Hardware overview of a single node of the DEEP-EST Extreme Scale Booster module [1].

- A **Data Analytics Module (DAM)** with 16 nodes, each equipped with two Intel Xeon 'Cascade Lake' Platinum 8260M CPU, clocked at 2.40GHz, an NVIDIA V100 Tesla GPGPU (32 GB HBM2), an Intel STRATIX10 FPGA (32 GB DDR4), 384 GB of RAM and 2 or 3 TB of non-volatile memory (14 nodes with 2 TB, 2 nodes with 3TB), two 1.5 TB Intel Optane SSDs (1 for local scratch, 1 for BeeOND), and 240 GB SSD for booting and the OS. The nodes are interconnected via an EXTOLL (100 Gb/s) network and a 40 Gb/s ethernet network. Figure 4 gives an overview of the corresponding node configuration.

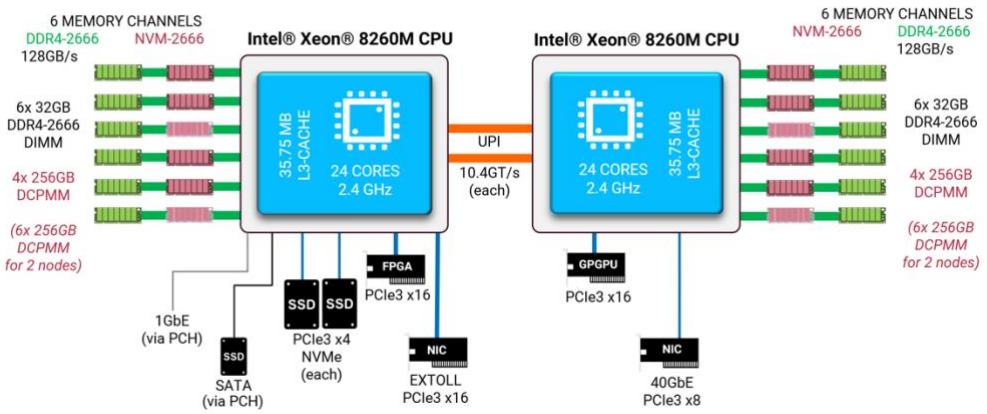

Figure 4: Hardware overview of a single node of the DEEP-EST Data Analytics Module [1].

In addition to the compute modules, the DEEP-EST system also holds further smaller test modules, which are also available to the user:

- A 4-node prototype DAM. Each node is equipped with two Intel Xeon 'Skylake' with 26 cores per socket and 192 GB of RAM. They are connected to a 1Gb/s ethernet network.
- A 16-node old DEEP-ER Cluster Module software development vehicle (SVD). Each node is equipped with two Intel Xeon 'Haswell' E5-v2680 v3, clocked at 2.5 GHz, 128 GB of RAM, an NVMe with 400 GB per node, which is accessible through BeeGFS on demand. The nodes are connected by a 100 Gb/s EXTOLL Tourmalet network.
- A 4-node Intel Knight's Landing (KNL) module, each consisting of an Intel Xeon Phi (64-68 cores), an NVMe with 400 GB per node, which is accessible through BeeGFS on demand, 16 GB of multi-channel dynamic random-access memory (MCDRAM) plus 96 GB of RAM per KNL. The nodes are interconnected by a Gb/s ethernet network.
- A 3-node GPU module for AI applications. Each node holds two Intel Xeon 'Skylake' Silver 4112 clocked at 2.6 GHz, 192 GB of RAM, four NVIDIA Tesla V100 GPU, connected via PCIe Gen3, and 16 GB HBM2. The interconnect is a 40Gb/s ethernet network.
- An old DEEP-ER NAM SDV with 2 GB and an EXTOLL interconnect[4].

2.2 Accessing the DEEP-EST system

To use computing resources at JSC, it is necessary to have a compute-time project. New users can join such a project to use the allocated resources. In the following, first, an overview of how to apply for resources at JSC is given in Sec. 2.2.1, before details on the creation of a user account, the assignment to a project, and a description on how to login to the system are given in Sec. 2.2.2, Sec. 2.2.3, and Sec. 2.2.4.

2.2.1 Compute-time applications

Principe investigators (PIs) with a sufficient scientific degree (Ph.D. and/or Prof.) can apply for computing time through open calls. However, for RAISE, access to the system will be provided through an already existing compute-time project, see Sec. 2.2.2.

[4] More information: https://www.deep-projects.eu/hardware/memory-hierarchies/49-nam

For completeness, the following options are available for users to apply for computing time at JSC.

- Applications for Large-Scale Projects of the Gauss Supercomputing Centre (GCS)[5] and the GCS and John von Neumann-Institute for Computing (NIC)[6] Regular Projects on JUWELS and/or JURECA Booster Module.
- Applications for Computing Time on JURECA via the Jülich Aachen Research Alliance (JARA)[7].
- Applications through the Partnership for Advanced Computing in Europe (PRACE)[8].
- Application for Test or Preparatory Access[9].

The proposals are reviewed technically and scientifically, and in the case of scientific excellence and proven code efficiency, computing time will be granted.

For more details, the reader is referred to the "Best practice guidelines/tutorials for MSA/heterogeneous systems" document of the CoE or JSC's website on compute-time applications[10], where the application process is explained in more detail.

2.2.2 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[11].

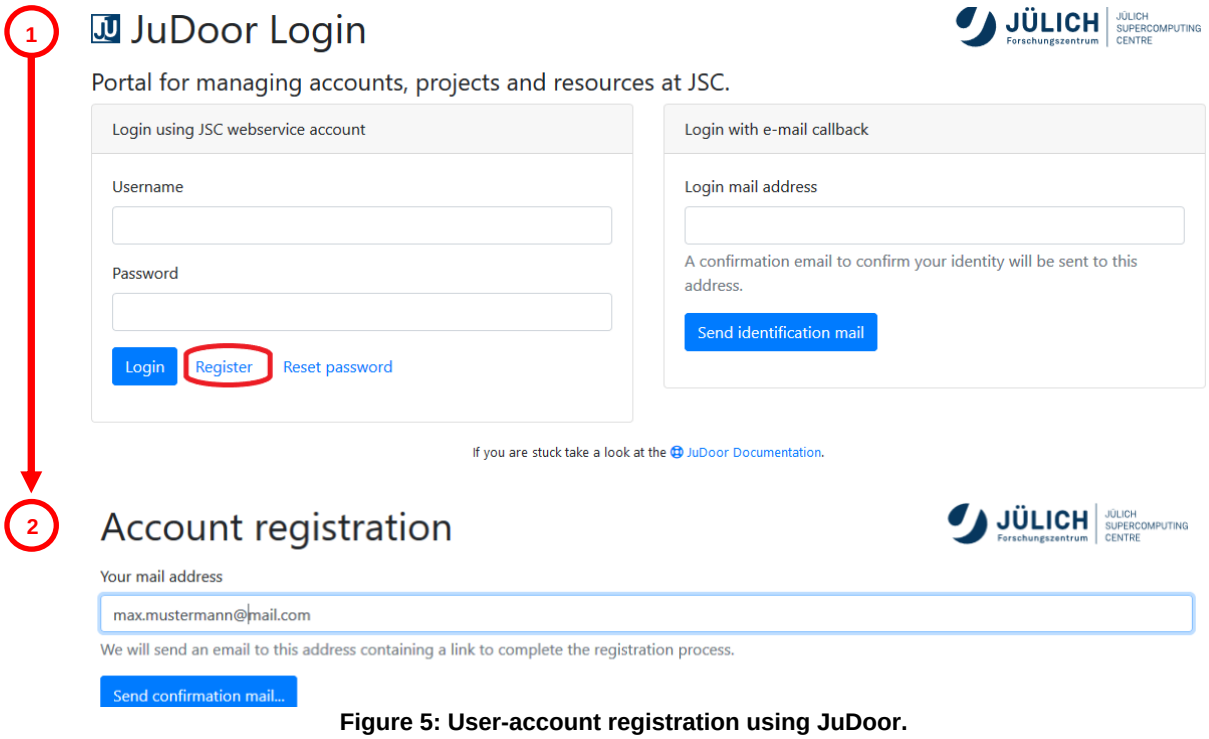

Figure 5: User-account registration using JuDoor.

[5] GCS https://gauss-centre.eu
[6] NIC http://www.john-von-neumann-institut.de
[7] JARA https://www.jara.org/en
[8] PRACE https://prace-ri.eu
[9] Preparatory Access https://www.fz-juelich.de/ias/jsc/EN/Expertise/SimLab/PrepAccess/PrepAccessNode.html
[10] How to apply for computing time https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/ComputingTime/computingTime_node.html
[11] JuDoor https://judoor.fz-juelich.de

The system allows managing personal user data and accounts on the HPC systems at JSC and the corresponding project memberships.

To create a new account on JSC's systems, it is necessary to register on JuDoor's login website, see Figure 5 – step 1. After clicking the "Register" button, the user must complete a form, where an email address needs to be provided (Figure 5 – step 2). An email containing a registration link is then sent to the user, where additional personal data needs to be provided. After completion, an account is created, and the user can now log in to the JuDoor system.

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 6 – step 3. This opens the form shown in Figure 6 – step 4. Here, a project id needs to be provided. The project id for a project on the DEEP-EST system, which can be used in the CoE is "deepext" ("TypeX_Y" in Figure 6). Additionally, further information can be provided. For RAISE, JSC asks the applying user to type the partner's name and the project acronym "RAISE" into this field. By clicking the "Join project" button, the user's request is sent to the PI or the project administrator (PA) of the corresponding project on the DEEP-EST system. The user then has to wait until the PI/PA grants access to resources on the system.

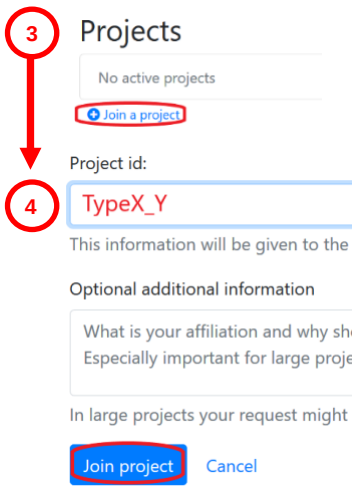

Figure 6: Joining a project in JuDoor.

After resources have been granted, new systems will appear in the section "Systems". For each of them, it is necessary to sign the usage agreement, see Figure 7.

Figure 7: Signing the usage agreement for accessible systems.

2.2.3 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 can also assign a 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, where the PI/PA can confirm the project membership and can assign available resources to the user.

2.2.4 Logging into the system

The DEEP-EST system is available via `ssh`. Before it is possible to start working, an `ssh` key needs to be uploaded. The procedure and the security constraints, together with how to login to the system, are described subsequently.

After the user agreement, see Sec. 2.2.2, has been signed, a new link “Manage SSH-keys” right next to the project appears. The corresponding website already provides some examples on how to fill the form, see Figure 8. Due to the European-wide security incident in 2020, the

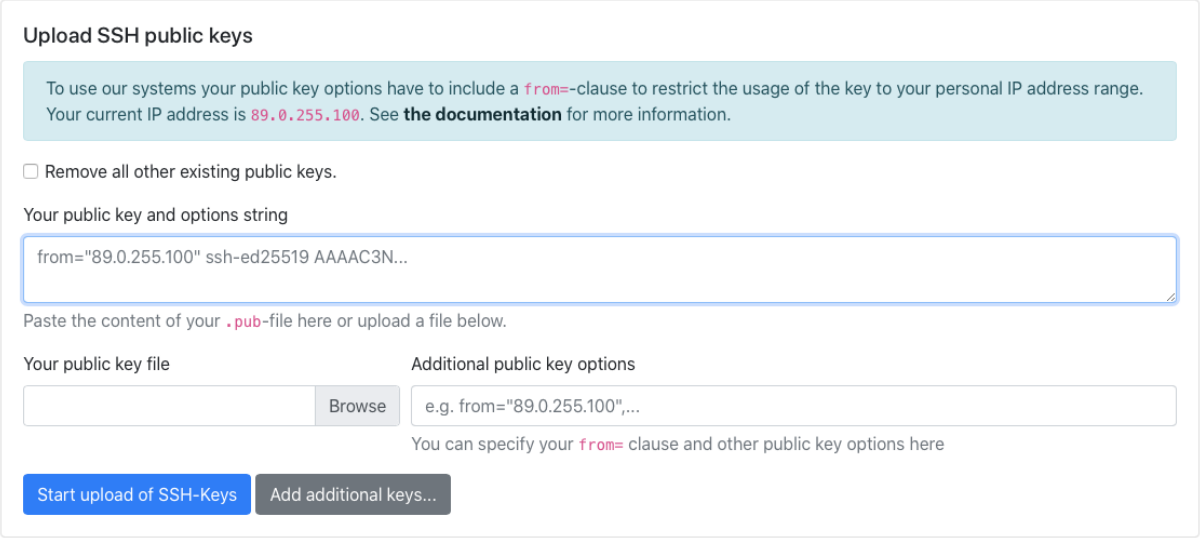

Figure 8: Upload form for ssh-keys in JuDoor.

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.

To generate an `ssh` key on a Linux or MacOS system, a terminal application needs to be opened, and the following command needs to be typed:

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

This will create a private/public key pair in the folder `~/.ssh/id_ed25519` with key type `ed25519`. In the generation process, the user will be asked to provide a password to protect the key. Here it is necessary not to leave the password empty.

The content of your 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” form or the key needs to be uploaded via the “Your public key file” field. In case a Windows-based OS is used, it is recommended to read the `ssh` tutorial on www.ssh.com[12], which explains how to use `Putty` as an `ssh` client and for the key pair generation.

The content of the from clause, the `89.0.255.100` in the example in Figure 8, 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 CIDR (Classless Inter-Domain Routing) notation, like `134.94.0.0/16`

[12] SSH Putty tutorial https://www.ssh.com/ssh/putty/windows/puttygen

For more information on how to generate keys and specify key option strings, the reader is referred to the JUWELS access documentation[13].

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

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

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

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

Then, the login command can be exchanged by

```
$ ssh deep
```

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

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

2.3 Using the DEEP-EST system

What makes the DEEP-EST system unique is that it is a prototype system that unites different kinds of hardware. With this heterogeneity also, the complexity of using the system increases. While some features such as the module system and the software stack are quite standard on the DEEP-EST system, other features are hardware-specific. In the following, the more standardized features, i.e., the software stack using the module system `Easybuild`, are introduced in Sec. 2.3.1. Subsequently, the more DEEP-EST specific features such as the file systems with different kinds of file system types (Sec. 2.3.2), important things to consider for software development on the DEEP-EST system (Sec. 2.3.3), the SLURM batch system with its special features for heterogeneous computation (Sec. 2.3.4), and DEEP-EST module-specific information (Sec. 2.3.5) are provided.

2.3.1 Software stack and introduction to the module system

The DEEP-EST system uses the `Easybuild` module system, which allows changing 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 following command

```
$ module list
```

[13] JUWELS access documentation https://apps.fz-juelich.de/jsc/hps/juwels/access.html

and 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

```
$ 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 particular 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
```

2.3.2 Available file systems

On the DEEP-EST system, three different groups of file systems are available:

- JSC's General Parallel File System (GPFS), provided via the Jülich Storage Cluster (JUST) and mounted on all JSC systems,
- the DEEP-EST (and SDV) parallel BeeGFS file systems, available on all the nodes of the DEEP-EST system,
- and the file systems local to each node.

The following Table 1 summarizes the characteristics of the file systems available in the DEEP-EST and DEEP-ER (SDV) systems.

Mount point	**Module**	**Type**
`/p/home`	SDV, DEEP-EST	GPFS exported via Network File System (NFS) protocol
`/p/project`	SDV, DEEP-EST	GPFS exported via NFS

Mount point	Module	Type
/arch	login node only (deepv)	GPFS exported via NFS
/work	DEEP-EST	BeeGFS
/scratch	DEEP-EST	Extended File System (XFS) local partition
/nvme/scratch	DAM	local SSD (XFS)
/nvme/scratch2	DAM	local SSD (Extended File System v4 - EXT4)
/pmem/scratch	DAM	DC Persistent Memory Modules (DCPMM) in appdirect mode
/sdv-work	SDV (deeper-sdv nodes via EXTOLL, KNL and ml-gpu via Gb/s ethernet only)	KNL nodes
/nvme	SDV	NVMe device
/mnt/beeond	SDV	BeeGFS on-demand running on the NVMe

Table 1: Available file systems on the DEEP-EST system.

2.3.3 Software development on the DEEP-EST system

It is recommended to use the Intel compiler and toolchain on the DEEP-EST system unless there are good reasons to move to the GCC or PGI compilers. To load the Intel compiler that has been compiled with GCC the following command needs to be used:

```
$ module load Intel/2019.3.199-GCC-8.3.0
```

(use module load gcc or module load pgi for the other compilers). If development for the NVIDIA GPGPUs is required, the CUDA environment can be loaded with

```
$ module load CUDA
```

The Intel oneAPI programming model simplifies the programming of CPUs and accelerators using modern C++ features to express parallelism with an open-source programming language called Data Parallel C++ (DPC++). The corresponding environment can be loaded via sourcing:

```
$ source /usr/local/intel/oneapi/inteloneapi/setvars.sh
```

For MPI-parallelized programs, it is recommended to use ParaStationMPI, which wraps by default the Intel compilers, i.e.,

```
$ mpicc --version
icc (ICC) 19.0.3.199 20190206 Copyright (C) 1985-2019 Intel Corporation.
All rights reserved.
```

The `ParaStationMPI` module can be loaded by calling

```
$ module load ParaSationMPI
```

Once loaded the wrappers `mpicc`, `mpicxx`, `mpif77`, and `mpif90` can be used to compile source code, e.g.,

```
$ mpicc a.c -o a.exe
$ mpicxx a.C -o a.exe
$ mpif77 a.f -o a.exe
$ mpif90 a.f -o a.exe
```

In case it is desired to not use the wappers, more information on the necessary include files and libraries can be obtained by providing the `-show` parameter to the MPI wrappers. For example, the output for the `C++` MPI wrapper is

```
$ mpicxx -show
  icpc -Wl,-rpath-link=/usr/local/.../pscom/Default/lib \
  -I/direct/usr_local/.../5.2.2-1-iccifort-2019.3.199-GCC-8.3.0/include \
  -L/direct/usr_local/.../5.2.2-1-iccifort-2019.3.199-GCC-8.3.0/lib \
  -lmpicxx -Wl,-rpath \
  -Wl,/direct/usr_local/.../5.2.2-1-iccifort-2019.3.199-GCC-8.3.0/lib \
  -Wl,--enable-new-dtags -lmpi
```

and for the `FORTRAN90` compiler

```
$ mpif90 -show
  ifort -reentrancy threaded \
  -Wl,-rpath-link=/usr/local/.../pscom/Default/lib \
  -I/direct/usr_local/.../5.2.2-1-iccifort-2019.3.199-GCC-8.3.0/include \
  -I/direct/usr_local/.../5.2.2-1-iccifort-2019.3.199-GCC-8.3.0/include \
  -L/direct/usr_local/.../5.2.2-1-iccifort-2019.3.199-GCC-8.3.0/lib \
  -lmpifort -Wl,-rpath \
  -Wl,/direct/usr_local/.../5.2.2-1-iccifort-2019.3.199-GCC-8.3.0/lib \
  -Wl,--enable-new-dtags -lmpi
```

2.3.4 Batch system on the DEEP-EST system

The DEEP-EST system uses the `SLURM` job scheduler[14] for batch system management. `SLURM` offers interactive and batch jobs (scripts submitted into the system). Jobs need to be allocated to one of the partitions listed in Table 2. Further information on `SLURM`'s environment variables and how to interpret job status and reason codes can be found in Annex A Additional information on SLURM.

Name	Nodes	Description
`dp-cn`	`dp-cn[01-50]`	DEEP-EST Cluster nodes (CNs)(Xeon Skylake)
`dp-dam`	`dp-dam[01-16]`	DEEP-EST DAM nodes (Xeon Cascadelake + 1 V100 + 1 Stratix 10)
`dp-dam-ext`	`dp-dam[09-16]`	DEEP-EST DAM nodes connected with Extoll Tourmalet
`dp-esb`	`dp-esb[01-25]`	DEEP-EST ESB nodes (Xeon Cascadelake + 1 V100)
`dp-sdv-esb`	`dp-sdv-esb[01-02]`	DEEP-EST ESB Test nodes (Xeon Cascadelake + 1 V100)
`ml-gpu`	`ml-gpu[01-03]`	GPU test nodes for ML applications (up to 4 V100 cards)
`sdv`	`deeper-sdv[01-16]`	cluster test nodes with Xeon Haswell CPU
`extoll`	`deeper-sdv[01-16]`	these nodes use an Extoll Tourmalet fabric
`knl`	`knl[01,04-06]`	KNL nodes
`knl256`	`knl[01,05]`	KNL nodes with 64 cores
`knl272`	`knl[04,06]`	KNL nodes with 68 cores
`snc4`	`knl[05]`	KNL node in snc4 memory mode
`psgw-cluster`		gateway test node
`psgw-booster`		gateway test node
`debug`		all compute nodes (no gateways)

Table 2: Available SLURM partitions on the DEEP-EST system.

[14] SLURM website https://slurm.schedmd.com

Information on partitions can be obtained by (replace `<partition>` with the partition name):

```
$ scontrol show partition <partition>
```

or simply by calling

```
$ sinfo
```

Interactive jobs

Interactive jobs can be started by first allocating nodes for execution

```
$ srun -A deep -p dp-cn -N 4 -n 8 -t 00:30:00 --pty /bin/bash -i
```

In this example, four nodes (`-N 4`) with two tasks per machine (a combination of `-N 4 -n 8`) on the partition `dp-cn` are requested for `30` minutes. An executable that uses all the requested resources can then be run by (replace `program` by the executable):

```
$ srun ./program
```

Both these steps can be combined by

```
$ srun -A deep -p dp-cn -N 4 -n 8 -t 00:30:00 ./program
```

Batch jobs

Jobs can also be submitted via a job script that holds all the 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=dp-cn
#SBATCH -A deep
#SBATCH -N 4
#SBATCH -n 8
#SBATCH -o /p/project/cdeep/user/hello_cluster-%j.out
#SBATCH -e /p/project/cdeep/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 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
```

Heterogeneous jobs

The MSA environment together with `SLURM` allows submitting jobs requesting heterogeneous hardware. The following asks for each one node with each one task per node from the partitions `dp-cn` and `dp-dam` with a specific request for nodes `dp-cn01` and `dp-dam01`:

```
$ srun --account=deep \
  --partition=dp-cn -N 1 -n 1 hostname :
  --partition=dp-dam -N 1 -n 1 hostname dp-cn01 dp-dam01
```

The same can be achieved via the `salloc` command:

```
$ salloc \
  --partition=dp-cn -N 1 -n 1 hostname : \
  --partition=dp-dam -N 1 -n 1 hostname dp-cn01 dp-dam01
```

Heterogeneous batch jobs

Job execution using heterogeneous architecture can also be triggered by a job script. In the following example script, a job with name `imb_execute_1` for the account `deep` with the first partition `dp-cn` and the additionally packed partition `dp-dam` is requested for `2` minutes. For both partitions, each a single node (`--nodes=1`) with 12 MPI (`--ntasks-per-node=12`) ranks each with a single task (`cpus-per-task=1`) is requested. Note that the `srun` command calls two executables `program-cn` and `program-dam` that have been individually compiled for the two different architectures.

```
#!/bin/bash
#SBATCH --job-name=imb_execute_1
#SBATCH --account=deep
#SBATCH --mail-user=
#SBATCH --mail-type=ALL
#SBATCH --output=job.out
#SBATCH --error=job.err
#SBATCH --time=00:02:00
#SBATCH --partition=dp-cn
#SBATCH --nodes=1
#SBATCH --ntasks=12
#SBATCH --ntasks-per-node=12
#SBATCH --cpus-per-task=1
#SBATCH packjob
#SBATCH --partition=dp-dam
#SBATCH --constraint=
...
```

```
...
#SBATCH --constraint=
#SBATCH --nodes=1
#SBATCH --ntasks=12
#SBATCH --ntasks-per-node=12
#SBATCH --cpus-per-task=1
srun ./program_cn : ./program_dam
```

When submitting a heterogeneous job with this colon notation using `ParaStationMPI`, a unique `MPI_COMM_WORLD` is created, spanning across the two partitions. If this is not desired, the last line can also be exchanged by

```
...
srun --pack-group=0 ./program_cn ; srun --pack-group=1 ./program_dam
```

Heterogeneous batch jobs that use MPI across modules

In order to establish MPI communication across modules using different interconnect technologies, some special gateway nodes must be used. On the DEEP-EST system, MPI communication across gateways is needed only between Infiniband and Extoll interconnects.

The following only works with `ParaStationMPI`:

```
#!/bin/bash
#SBATCH --job-name=modular-imb
#SBATCH --account=deep #SBATCH --time=00:10:00
#SBATCH --output=modular-imb-%j.out
#SBATCH --error=modular-imb-%j.err
#SBATCH --gw_num=1
#SBATCH --gw_psgwd_per_node=1
#SBATCH --partition=dp-cn
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1  #SBATCH packjob
#SBATCH --partition=dp-dam-ext
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
echo "DEBUG: SLURM_JOB_NODELIST=$SLURM_JOB_NODELIST"
echo "DEBUG: SLURM_NNODES=$SLURM_NNODES"
echo "DEBUG: SLURM_TASKS_PER_NODE=$SLURM_TASKS_PER_NODE"
module --force purge
module use $OTHERSTAGES
module load Stages/Devel-2019a
module load Intel
module load ParaStationMPI
srun hostname : hostname
APP="./IMB-MPI1 Uniband"
srun ${APP} : ${APP}
```

2.3.5 Module-specific information

Each of the different DEEP-EST modules is unique and therefore different to use. In the following, the module-specific features are explained and how they can be used to develop software and to execute code. Note that all necessary information to use the CN module is already covered in Sec. 2.3.4.

DAM module

Each of the DAM nodes is equipped with Intel's Optane® DCPMM. Whereas the first two nodes (`dp-dam[01,02]`) expose 3 TB of persistent memory the remaining nodes (`dp-dam[03-16]`) have 1.5 TB of persistent memory included. The DCPMMs can be driven in different modes. To check which nodes are running in which mode, the user can use the `scontrol` command:

```
$ scontrol show node dp-dam01 | grep AvailableFeatures
```

To select a node using a certain memory mode, the user can use the constraint option within `SLURM`, e.g.

```
$ srun -p dp-dam -N 1 -c 24 -t 0:30:0 \
  --constraint=dpcmm_mem --pty /bin/bash
```

Each node is equipped with a Stratix 10 Field Programmable Gate Array (FPGA). For getting started using OpenCL (Open Computing Language) with the FPGAs, some hints and slides and exercises from the Intel FPGA workshop held at JSC can be found in `/usr/local/fpga`. It is recommended to do the first steps in an interactive session on a DAM node. To set up and check the FPGA environment, the following steps need to be followed:

```
$ source /usr/local/fpga/FPGA_init.sh
$ lspci | grep -i 'accel'
$ aocl list-devices
$ aoc -list-boards
# -- optional for doing the exercises:
# export CL_CONTEXT_EMULATOR_DEVICE_INTELFPGA=1
```

The lab can be copied and untared into the user's home directory to do the exercises step by step. The exercises use the emulator device instead of the actual FPGA device due to the FPGAs' long compilation time. For using the FPGA device, the OpenCL kernels need to be compiled using the `-board=pac_s10_dc` option:

```
# compile for the emulator
$ aoc -march=emulator -fast-emulator kernel-file.cl
# compile for the FPGA device
$ aoc -board=pac_s10_dc kernel-file.cl
```

In addition, the OpenCL host file to select the correct platform ("Intel® FPGA SDK for OpenCL™" or "Intel® FPGA Emulation Platform for OpenCL™ (preview)") needs to be adapted.

ESB module

The ESB module is equipped with NVIDIA Tesla V100 GPGPUs. To use GPU direct over Infiniband on the ESB when passing GPU buffers directly to `MPI_send` and `MPI_receive` functions, the following modules have to be loaded. Note that it is planned to later integrate those modules in the production stage of `Easybuild` such that only loading the `Intel` and `ParaStationMPI` modules will be required:

```
$ module --force purge
$ module use $OTHERSTAGES
$ module load Stages/Devel-2019a
$ module load Intel
$ module load ParaStationMPI
```

In addition, the following environment settings are required for the job scripts or interactive sessions:

```
$ export PSP_CUDA=1
$ export PSP_UCP=1
```

SDV cluster module

In the SDV cluster, NVMe is available. During job startup, all files of non-privileged users within `/nvme` are removed. If files need to be kept across consecutive jobs on a particular SDV node, a list of filenames needs to be added to `~/.nvme_keep`, e.g.,

```
/nvme/tmp/myfile.A
/nvme/tmp/myfile.B
```

The SDV cluster nodes are connected via EXTOLL Tourmalet (100 Gb/s). Therefore, the following needs to be loaded to run jobs on multiple nodes:

```
$ module load extoll
```

KNLs

When compiling code for the KNLs, the `-xMIC-AVX512` flag instead of the `-mmic` needs to be provided. It is highly recommended to use the fast MCDRAM of the KNLs when executing code. The MCDRAM can either be in cache or flat mode. In the former case, no changes are needed to utilize the MCDRAM. In the latter case, two subcases exist:

- If the total memory footprint of the application is smaller than the size of the MCDRAM use (note that allocations that don't fit into MCDRAM make the application fail):

  ```
  $ numactl –m 1 ./program
  ```

- If the total memory footprint of the application is larger than the size of the MCDRAM use (note that allocations that don’t fit into MCDRAM spill over to DDR):

```
$ numactl –p 1 ./program
```

3 JUAWEI prototype at Forschungszentrum Jülich

The JUAWEI system is a small HUAWEI prototype installed at JSC featuring ARM-based low-energy hardware technologies for testing and benchmarking purposes. In what follows, Sec. 3.1 delivers a description of the hardware specifications of the JUAWEI prototype. Section 3.2 explains how to access the system and Sec. 3.3 how to use the system.

A Wiki can be found on JSC's trac website[15]. The user support is available by email (sc@fz-juelich.de).

3.1 JUAWEI system hardware specifications

The JUAWEI prototype comprises Huawei Taishan 2280 servers based on Hisilicon ARM processors, i.e., Hi1610, Hi1612, and Hi1616 CPUs, where only the latter is available to the RAISE project. Furthermore, there are also Intel Haswell servers available as reference. In more detail, the system consists of the following components:

- Twelve Huawei Taishan 2280 nodes, each equipped with two Hi1616 CPUs with 32 Cortex-A72 cores each, clocked at 2.4GHz, and 256GB (16x16GB) registered DDR4 RAM, clocked at 2133MHz. The nodes are interconnected with an optical 10Gb/s ethernet with a random direct memory access (RDMA) over Converged Ethernet (RoCE) protocol.
- Ten Huawei RH2288HV3 nodes, each equipped with two Intel Xeon E5-2660 v3 (10 cores per CPU), clocked at 2.6Ghz (3.3GHz Boost) with Haswell cores, and 128GB (8x16GB) of registered DDR4 RAM, clocked at 2133MHz. The nodes are interconnected via an Infiniband network.

Both node types operate a CentOS 7.6.1810.

Furthermore, the system is currently extended by two AI nodes, i.e., an Atlas training node (Atlas 800, model 9000) and an Atlas inference node (Atlas 800, model 3000). They are, however, not yet in operation.

3.2 Accessing the JUAWEI system

As JUAWEI is a prototype, its resources are not accessible via the standard compute time applications. However, access can be granted for the members of RAISE upon request. Therefore, applicants need to follow the subsequent steps explained in Sec. 3.2.1 and Sec. 3.2.2.

3.2.1 User-account creation

To access the JUAWEI system, the steps described in Sec. 2.2.2 need to be followed. In contrast to the DEEP project, the user has to join the project "cjuawei00". For RAISE, JSC asks the applying user to type the partner's name and the project acronym into the corresponding field for further information. The PI/PA of the project will then grant access to the system.

[15] JUAWEI Wiki https://trac.version.fz-juelich.de/armlab/wiki/Public/Juawei

3.2.2 Logging into the system

Similar to the DEEP system, the JUAWEI system can only be reached via `ssh`. To access the system, a single frontend system is available, which can be reached via

```
$ ssh -i ~/.ssh/id_ed25519 user@juawei.zam.kfa-juelich.de
```

where `~/.ssh/id_ed25519` is the private key file and `user` is the username.

For further information concerning key pair generation and how to configure `ssh` for easy usage, the reader is referred to Sec. 2.2.4.

3.3 Using the JUAWEI system

In the following, an introduction on how to use available software stack on JUAWEI is given in Sec. 3.3.1, before some information on how to perform software development is provided in Sec. 3.3.2. Subsequently, the batch system on JUAWEI is explained in Sec. 3.3.3. It should be noted that since the JUAWEI system is a prototype, the software stack, the batch partitions, and the available hardware are subject to changes.

3.3.1 Software stack and module system

The JUAWEI system also uses `Easybuild` to organize its software stack, cf. Sec. 2.3.1. This stack more closely resembles what is available on other JSC systems, like JURECA and JUWELS. The `easyconfig` files used to build the software, are based on JSC's public `easyconfig` repository[16], but were modified to include support for the AArch64 architecture and the ARM HPC compiler.

The software stack is not enabled by default. To switch to the new software stack, the file `/opt/ohpc/pub/easybuild/switch_to_eb_sw_stack.sh` needs to be sourced via

```
$ source /opt/ohpc/pub/easybuild/switch_to_eb_sw_stack.sh
```

Please note: Due to the number of modifications/fixes that are required in some cases, not all software that is available on JURECA/JUWELS is available on JUAWEI!

3.3.2 Software development on the JUAWEI system

The `Easybuild` modules are organized in such a way that loading a compiler and an MPI environment will make all modules built with that compiler and MPI available, i.e., with

```
$ module load GCC OpenMPI
```

the user switched to the `GCC` and OpenMPI-based software stack. The available compilers, their status, and if they are available on the AArch64 and/or on the x86_64 nodes, is displayed in Table 3. For the MPI environment, only OpenMPI 4.0.1 is available.

[16] JSC easyconfig repository https://github.com/easybuilders/JSC

Name	Module	State	AArch64	x86_64
GCC	GCC/9.2.0	Full software stack	X	X
Clang	Clang/8.0.1	Compiler and select packages only	X	X
Clang	Clang/9.0.0	Compiler only	X	X
ARM-HPC	armhpc/19.2	Full software stack	X	O
ARM-HPC	armhpc/19.3	Full software stack	X	O

Table 3: Available compilers on the JUAWEI prototype.

Should the software be missing from the output of the command `module avail`, the user might need to delete the `lmod` cache by:

```
$ rm -rf ~/.lmod.d/.cache
```

The build systems and scripts in the software stack (`autotools`, `pkgconfig`, `CMake`, `Meson`, etc.) should find any software in the software stack and provide support for the particular build tool.

If pure `Makefiles` or another system are used, hardcoded paths should be avoided. Instead, the `Easybuild`-provided environment variables `$EBROOT<software name>`, which point to the root of the installed software, should be used, e.g., for OpenBLAS (Open Basic Linear Algebra Subprograms) the variable `$EBROOTOPENBLAS`.

Please note: Only a minimal level of support can be provided. In case a different version of a software or specific compile flags are required, it is probably easiest to have them compiled by the user him- or herself.

To use the MPI wrappers for `C`, `C++`, `FORTAN77`, or `FORTRAN90`, the following commands can be used:

```
$ mpicc a.c -o a.exe
$ mpicxx a.C -o a.exe
$ mpif77 a.f -o a.exe
$ mpif90 a.f -o a.exe
```

In case it is desired to not use the MPI wrappers, the necessary includes and library paths can be optained by providing the `-show` parameter to the corresponding compiler wrapper. For example, using `GCC`, this yields

```
$ mpicxx -show
  g++ -I/opt/ohpc/pub/.../OpenMPI/4.0.3-GCC-10.1.0/include \
  -L/opt/ohpc/pub/.../hwloc/2.1.0-GCCcore-10.1.0/lib \
  -L/opt/ohpc/pub/.../UCX/1.7.0-GCCcore-10.1.0/lib \
  -L/usr/lib64 -Wl,-rpath \
  -Wl,/opt/ohpc/pub/.../hwloc/2.1.0-GCCcore-10.1.0/lib \
  -Wl,-rpath Wl,/opt/ohpc/pub/.../UCX/1.7.0-GCCcore-10.1.0/lib \
  -Wl,-rpath -Wl,/usr/lib64 \
  -Wl,-rpath -Wl,/opt/ohpc/pub/.../OpenMPI/4.0.3-GCC-10.1.0/lib \
  -Wl,--enable-new-dtags \
  -L/opt/ohpc/pub/.../OpenMPI/4.0.3-GCC-10.1.0/lib \
  -lmpi
```

for `C++` and

```
$ mpif90 -show
  gfortran -I/opt/ohpc/pub/.../OpenMPI/4.0.3-GCC-10.1.0/include \
  -I/opt/ohpc/pub/.../OpenMPI/4.0.3-GCC-10.1.0/lib \
  -L/opt/ohpc/pub/.../hwloc/2.1.0-GCCcore-10.1.0/lib \
  -L/opt/ohpc/pub/.../UCX/1.7.0-GCCcore-10.1.0/lib \
  -L/usr/lib64 -Wl,-rpath \
  -Wl,/opt/ohpc/pub/.../hwloc/2.1.0-GCCcore-10.1.0/lib
  -Wl,-rpath -Wl,/opt/ohpc/pub/.../UCX/1.7.0-GCCcore-10.1.0/lib \
  -Wl,-rpath -Wl,/usr/lib64 -Wl,-rpath \
  -Wl,/opt/ohpc/.../OpenMPI/4.0.3-GCC-10.1.0/lib \
  -Wl,--enable-new-dtags \
  -L/opt/ohpc/pub/.../OpenMPI/4.0.3-GCC-10.1.0/lib \
  -lmpi_usempif08 -lmpi_usempi_ignore_tkr -lmpi_mpifh -lmpi
```

for `FORTRAN90`.

3.3.3 Batch system on the JUAWEI system

Like the DEEP system, the JUAWEI prototype uses `SLURM` for batch job processing. For an introduction on how to use `SLURM`, the reader is referred to Sec. 2.3.4. On JUAWEI, at present, two partitions as listed in Table 4 are available for submission.

Name	Description
`x86-normal`	JUAWEI x86_64-based nodes (Intel Haswell)
`arm-hi1616`	JUAWEI AArch64-based nodes (Hislicon Hi1616)

Table 4: Available partitions on the JUAWEI prototype.

Please note: This information is subject to continuous changes. To get details on the available partitions, it is recommended to run the command

```
$ sinfo
```

before submitting to the queuing system.

To run jobs on these partitions, the user may invoke the following commands to run an executable (`program`):

```
$ srun -p x86-normal program
$ srun -p arm-hi1616 program
```

To ensure that all users can run their compute jobs without issues and get accurate benchmark results, here is a couple of rules for the usage of the JUAWEI system:

- Any compute- and/or I/O-intensive job should be run through `SLURM` (`sbatch/srun`). `SLURM` will reserve a node for the job, and it will run it without suffering from side-effects of other running CPU- or I/O-heavy processes. This is especially important for running benchmarks, where accurate results are required.
- It is forbidden to run compute tasks on the login node (`juawei`).
- It is forbidden to directly login to the compute nodes via `ssh` and run heavy tasks, as this might interfere with scheduled tasks on that node. Only submissions via `SLURM` are allowed.
- Jobs that use all nodes should be avoided unless it is necessary. In this case, the job will stall until all nodes are available and will lock out all other users while it's running
- There are dedicated development nodes for x86_64 (`juawei-x12`) and AArch64 (`juawei-a28`) available, which should be used for code development/interactive sessions/compilation. That is, it is forbidden to use the login node (`juawei`) for development.
- Long compilations should be run through `SLURM`.
- The directories `/scratch` or `/tmp` should be used for heavy I/O. The `$HOME` directories share a 1Gb/s connection and overloading it will make the system unresponsive for everyone. If `/tmp` is used for I/O, it should be cleaned at the end of the scheduled job.

4 CTE-ARM prototype at Barcelona Supercomputing Center

The CTE-ARM prototype was installed very recently at BSC and is based on an ARM architecture, cf. the JUAWEI system at FZJ presented in Sec. 3. The system provides high performance, high scalability, and high reliability at ultra-low power consumption. Hence, it is of particular interest to the developers in RAISE as it provides a platform to benchmark and test RAISE's novel AI technologies and is a potential prototype for the next-generation exascale HPC systems. This BSC prototype will be available in 4-6 months to the RAISE project partners, i.e., between 06/2021 and 08/2021.

Subsequently, an overview of CTE-ARM's hardware specifications is given in Sec. 4.1. This is followed by an explanation on how to access the system in Sec. 4.2. Finally, Sec. 4.3 provides information on how to use the system. Continuously updated information is available from the CTE-ARM User's Guide[17].

4.1 CTE-ARM system hardware specifications

The CTE-ARM system features two login nodes and 192 compute nodes. The login nodes are equipped with an Intel(R) Xeon(R) Silver 4216 CPU, clocked at 2.10GHz and 256 GB of main memory. In contrast to the compute nodes, the login nodes are x86_64-based, i.e., it is necessary to cross-compile code to be executed on the compute nodes, which are ARM-based.

Each of the compute nodes has the following configuration:

- A64FX CPU (Armv8.2-A + SVE), clocked at 2.20GHz (4 sockets and 12 cores per socket; in total, there are 48 cores per node)
- 32GB of HBM2 main memory
- Single port Infiniband EDR
- Tofu Interconnect D network (TofuD) [2]

The system features the ARMv8.2-A Scalable Vector Extension (SVE) single instruction, multiple data (SIMD) instruction set with 512-bit vector implementation. The theoretical peak performance is 648.8TFlops in double precision, and its total amount of memory is 6TB, distributed across all nodes with each node featuring 32GB RAM. The system overview can be found on the User's Guide website.

4.2 Accessing the CTE-ARM system

Accessing the CTE-ARM system is quite simple. In the following, the method on how to apply for an account is described in Sec. 4.2.1, before explanations on how to login and change the password are given in Sec. 4.2.2.

4.2.1 Account application

As this system is a prototype, it is at present not open for the wider public. Therefore, access is currently limited to the BSC's direct project partners in RAISE, who can gain access through the following steps:

[17] CTE-ARM Users's Guide https://www.bsc.es/user-support/arm.php

1. Send an email to Guillaume Houzeaux (guillaume.houzeaux@bsc.es) or Oriol Lehmkuhl (oriol.lehmkuhl@bsc.es) from BSC and ask for system access.
2. Once the petition is accepted (generally within hours or a few days), the user's responsibilities agreement needs to be signed. Guillaume Houzeaux or Oriol Lehmkuhl will point the applicant to the right document.
3. Finally, the applicant will receive an email with all further instructions.

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

4.2.2 Login and password management

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

- `armlogin1.bsc.es`
- `armlogin2.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.

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.

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.

4.3 Using the CTE-ARM system

In the following, it is described how to use the CTE-ARM system at BSC. The available file systems and the data transfer mechanisms are introduced in Sec. 4.3.1. Subsequently, Sec. 4.3.2 briefly describes how to do software development. Finally, Sec. 4.3.3 gives an overview of how to use the batch system of the CTE-ARM system.

4.3.1 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 3 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 **Fujitsu Exabyte File System (FEFS)** is a scalable cluster file system based on Lustre with high reliability and high availability on all cluster nodes.
- The **GPFS** is a distributed networked file system with two partitions on CTS-ARM, i.e., `gpfs_home` and `gpfs_projects`. Both can be accessed from the login nodes and from the data transfer machine (see below). The export `gpfs_home` is also mounted on the compute nodes.

The parallel filesystems FEFS and GPFS allow 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.

The following Table 5 holds information on the available file systems on CTS-ARM.

Mount point	Soft link	Description
`/apps`	`/fefs/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`	`/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:** Do not run jobs from this file system. Instead, jobs should be executed from the group's `/scratch`.
`/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 it is distributed or shared

Mount point	Soft link	Description
		between the users. This file system is not mounted on the computing nodes.
/scratch	/fefs/scratch	This is the only file system intended for executions. For every group there exists a group directory with subdirectories for each user. It is mounted on the login and compute nodes. It is necessary to transfer all required scripts, input files, and any other kind of data to this file system before launching a job.

Table 5: Available file systems on CTS-ARM

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

As explained in Sec. 4.2.2, 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>
```

- SSHFTP

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

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 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`:

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

- `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`:

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

- `dtmv` submits an `mv` command to queues:

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

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 in Sec. 4.2.2, outgoing internet connection are not allowed from the cluster. This prevents the use of external repositories directly from the CTS-ARM 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 will directly be 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
- 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`.

4.3.2 Software development on the CTE-ARM system

Standard compiler wrappers for `Fortran`, `C`, and `Java`, all with MPI are available, e.g.,

```
$ mpifrtpx tesf.f -o test
$ mpifccpx test.c -o test
```

For longer compilations, it is recommended to request a node interactively or to submit a job script and run the compilation on the cluster. Examples on how to submit jobs can be found in the subsequent Sec. 4.3.3.

4.3.3 Batch system on the CTE-ARM system

`PJM` is the utility used for batch processing support, i.e., all jobs must be run through it. This section provides information for getting started with job execution on the cluster.

Job queues

There are several queues configured in `PJM`, and different users may access different queues. To list all queues the user has access to, the following command can be used:

```
$ pjshowrsc --rg
[ CLST: compute ]
[ RSCUNIT: rscunit_ft02 ]
RSCGRP           NODE
                 TOTAL  CNS FREE  ALLOC
small               24     24      0
middle              96     96      0
important          192    192      0
def_grp             96     96      0
large              192    192      0
```

Job submission

A job is the execution unit of `PJM`. A job file is defined by a text file containing a set of directives describing the job's requirements and the commands to execute.

The following is the basic directives to submit a job with file name `<job_script>`:

```
$ pjsub <job_script>
```

To shows all the submitted jobs, the command

```
$ pjstat
```

can be used. To hold and release a non-empty set of jobs with given job id `<job_id>`, the following commands can be executed:

```
$ pjhold <job_id>
$ pjrls <job_id>
```

To delete a job with id `<job_id>`, the command

```
$ pjdel <job_id>
```

is used. More details on these commands can be obtained from the corresponding `man` pages.

Disclaimer about job submissions

User, who are accommodated to using the other HPC clusters at BSC, should note that there's a big difference that needs to be taken into account when using the CTE-ARM system. In this cluster, in order to avoid potential issues when trying to write or access files at job execution time, **it is imperative** that the output files and the working directories are located inside the `/fefs` file system. This also includes the paths of the job output/error files. Failing to do so may make the jobs fail unexpectedly.

Interactive sessions

Allocation of an interactive session has to be done through `PJM` by:

```
$ pjsub --interact
```

Job directives

A job must contain a series of directives to inform the batch system about the characteristics of the job. These directives appear as comments in the job script, here you may find the most common directives for both syntaxes.

Specify the name of the job:

```
#PJM -N <name>
```

Store both standard output and standard error to the same file (this will ignore the `-e` directive if specified):

```
#PJM -j
```

Inherit environmental variables at batch job submission to the running environment of the job:

```
#PJM -X
```

Name of the resource group to submit the job to (replace `<name>` by a matching group):

```
#PJM -L rscgrp=<name>
```

The limit of wall clock time, which needs to be set to a value greater than the real execution time for the application (the job will be killed after the time has passed):

```
#PJM -L elapse=[[HH:]MM:]SS
```

The number (`<number>`) of requested nodes:

```
#PJM -L node=<number>
```

The following option specifies the parameters of an MPI job:

```
#PJM --mpi "parameter[...]"
```

These are more common parameters:

- `proc=<number1>`: the number (`<number1>`) of processes to start
- `max-proc-per-node=<number2>`: the number (`<number2>`) of processes by node

To use both options simultaneously and to tune the MPI setting for the job, the following syntax has to be used:

```
#PJM --mpi "proc=<number1>,max-proc-per-node=<number2>"
```

The name of the file (`filename`) to collect the standard output (`stdout`) of the job:

```
#PJM -o filename
```

The name of the file to collect the standard error output (`stderr`) of the job:

```
#PJM -e filename
```

Standard output/error management

Standard output and standard error output are saved in files. If the output files are not specified, they will be created in the directory where the `pjsub` command was issued (`%n` is the job name, which is the name of the job script if not specified, and `%j` is the job id):

- `%n.%j.out`: standard output
- `%n.%j.err`: standard error output

Job file examples

An example of a sequential job running for `5` minutes in the partition `small` the command `/usr/bin/hostname` could look as follows:

```
#!/bin/bash

#------ pjsub option --------#
#PJM -L "rscgrp=small"
# Name of the resource group (= queue) to submit the job
#PJM -N serial
# Name of the job (optional)
#PJM -L node=1
# Specify the number of required nodes
#PJM -L elapse=00:05:00
# Specify the maximum duration of a job
#PJM -j
# Store stdout and stderr in the same file

#------- Program execution -------
# /usr/bin/hostname
```

The job is submitted using the `pjsub` command. The output will be stored in the same directory as the file `serial.*.out`, where `*` is the job id.

An example of a job file for a parallel execution of the program `/fefs/apps/examples/text` using `6` MPI ranks distributed over `2` nodes and running for `30` minutes could look as follow:

```
#!/bin/bash
#PJM -N parallel
#PJM -L rscgrp=small
#PJM -L node=2
#PJM -L elapse=0:30:00
#PJM --mpi "proc=6,max-proc-per-node=3"
# The number of MPI processes and the maximum of processes per node
...
```

```
...
#PJM -o job-%j.out
# File where standard output will be stored
#PJM -e job-%j.err
# File where standard errors will be stored

export PATH=/opt/FJSVxtclanga/tcsds-1.1.18/bin:$PATH
export LD_LIBRARY_PATH=/opt/FJSVxtclanga/tcsds- \
                       1.1.18/lib64:$LD_LIBRARY_PATH

mpirun -np 6 /fefs/apps/examples/test
```

5 CTE-AMD prototype at Barcelona Supercomputing Center

The CTE-AMD system is a supercomputer based on AMD Epyc processors. The AMD processor families have gained massive popularity over the last years not only on the consumer market but also on the server marked. They have become the direct competitors of the Intel processor families as they allow for higher parallelism at lower prices. Hence, they are also subject to investigations in RAISE as they are also candidates to be integrated into the next-generation exascale systems.

Like the CTE-ARM system, the CTE-AMD system has been installed recently at BSC and will be available in 4-6 months to the RAISE partners, i.e., between 06/2021 and 08/2021.

Continuously updated information on the system can be found in BSC's CTE-AMD User's Guide[18]. The CTE-AMD system hardware specifications are discussed next.

In the following, Sec. 5.1 provides more details on the CTE-AMD hardware specifications. Subsequently, Sec. 5.2 explains how to access the system. A usage guide is presented in Sec. 5.3.

5.1 CTE-AMD system hardware specifications

The CTE-AMD system is operated with a Linux Operating System, i.e., CentOS Linux release 8.1.1911 (Core). There exist one login and 33 compute nodes. They are interconnected via an Infiniband network. In more detail, each node is configured as follows:

- A single AMD EPYC 7742 CPU, clocked at 2.25GHz, with 64 cores and 2 threads/core, i.e., in total 128 threads per node are possible
- 1,024GB of main memory distributed over 16 DIMMS a 64GB, clocked at 3200MHz
- A 480GB SSD as local storage
- Two AMD Radeon Instinct MI50 GPGPUs equipped with 32GB RAM
- A single Port Mellanox Infiniband HDR100
- GPFS is connected via two 10Gb/s copper links

5.2 Accessing the CTE-AMD system

Gaining access to the CTE-AMD system is similar to gaining access to the CTE-ARM system, see Sec. 4.2. There are, however, some smaller differences worth mentioning. The following Sec. 5.2.1 describes how to apply for an account. Section 5.2.2 explains on how to login and change the password.

5.2.1 Account application

Similar to the CTE-ARM system, this system is a prototype, i.e., it is at present not open for the wider public. Therefore, access is currently limited to BSC's direct project partners in RAISE, who can gain access through the same steps as described in Sec. 4.2.1. The reader is referred to this section to get details on how to apply for an account.

[18] CTE-AMD User's Guide https://www.bsc.es/user-support/amd.php

5.2.2 Login and password management

A public login node is available to connect to the CTE-AMD system:

- `amdlogin1.bsc.es`

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

The `ssh` tools need to be used to login into the system and to transfer files into 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. The method to change the password is described in Sec. 4.2.2. Note that the password change takes about 10 minutes to be effective.

5.3 Using the CTE-AMD system

In the following, the software stack is first described and an introduction to the module system is given in Sec. 5.3.1. The available file systems and the data transfer mechanisms are introduced in Sec. 5.3.2. Section 5.3.3 explains how to do software developments on the system and Sec. 5.3.4 describes how to use the batch system on the CTE-AMD cluster.

5.3.1 Software stack and introduction to the module system

The current software stack is the GCC (Red Hat) 8.3.1–4, Radeon Open Compute (ROCm) Runtime software stack 3.5.0.

Modules Environment

The `Environment Modules` package[19] 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-AMD 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 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.

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

- **Compilers:** Contains various compiler suites available for the system (`Intel`, `GCC`, etc.), see also Sec. 5.3.3.

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 `GCC` module solely by its name:

```
$ module load gcc
```

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

```
$ module load gcc/10.2.0
```

The most important commands for using the module system are, (cf. Sec. 2.3.1 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 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 options, or to check the `module(1) manpage` 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/resources 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 for all machines). More information can be obtained by:

```
$ man <bsc_command>
```

Important note: Some of these commands might not be available yet. The machine is fairly new and some commands are still being ported.

5.3.2 Available file systems and data transfer

IMPORTANT: It is in the user's responsibility to backup all your critical data. BSC only guarantees a daily backup of user data under `/gpfs/home`. Any other backup should only be done exceptionally under the demand of 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 3 different types of storage available in 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 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 6 holds information on the available file systems on CTS-AMD.

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:** Do not run jobs 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

Mount point	Description
	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 it is distributed or shared between their users. This file system is not mounted on the computing nodes.
`/gpfs/scratch`	Each user will have a directory on `/gpfs/scratch`. Its intended use is to store temporary files of the jobs during their execution. A quota per group will be 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 will not be available from the login nodes.

Table 6: Available file systems on CTS-AMD.

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. 5.3.1):

```
$ bsc_quota
```

The command provides a readable output for the quota. **Please note:** The command might still not be available, as the machine is fairly new and the usual `bsc_commands` are still being ported.

If more disk space on any file system of CTE-AMD 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.

Data transfer

The data transfer mechanisms for the CTE-AMD system are the same as for the CTE-ARM system. The reader is therefore referred to Sec. 4.3.1 for examples on how to use direct transfer methods, the data transfer machines, the AA management, and GIT/SVN with `sshfs`.

5.3.3 Software development on the CTE-AMD system

All software and numerical libraries available on the cluster can be found in `/apps/`. In case something is missing, please contact BSC's support (support@bsc.es).

C/C++ compilers

On the CTE-AMD machine, the following C/C++ compilers (along others) are available:

- clang: This is an AMD-optimized compiler for C/C++ (from AOCC). Help can be obtained from:

```
$ man clang
```

- 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/C++ compilers follow the suffix conventions for input files listed in Table 7. 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 7: Suffix conventions for C/C++.

The default sizes of the standard C/C++ datatypes on the CTE-AMD machine are given in Table 8.

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 8: Default C/C++ datatype sizes.

The GCC provided by the system is version 8.3.1. For better support of new and old hardware features, different versions can be loaded via the provided modules. For example, on the CTE-AMD machine, GCC 10.2.0 can be loaded via

```
$ module load gcc/10.2.0
```

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:

```
$ module load openmpi
```

or

```
$ module load ibm_mpi
```

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

```
$ gcc -I/apps/OPENMPI/4.0.5/GCC/include/openmpi \
  -I/apps/OPENMPI/4.0.5/GCC/.../hwloc/include \
  -I/apps/OPENMPI/4.0.5/GCC/.../libevent \
  -I/apps/OPENMPI/4.0.5/GCC/.../libevent2022/libevent/include \
  -I/apps/OPENMPI/4.0.5/GCC/include \
  -pthread -Wl,-rpath -Wl,/apps/OPENMPI/4.0.5/GCC/lib \
  -Wl,--enable-new-dtags -L/apps/OPENMPI/4.0.5/GCC/lib -lmpi
```

For C++, the command is

```
$ mpicxx -show
```

which results in the following output

```
$ g++ -I/apps/OPENMPI/4.0.5/GCC/include/openmpi \
  -I/apps/OPENMPI/4.0.5/GCC/.../hwloc/include \
  -I/apps/OPENMPI/4.0.5/GCC/.../libevent \
  -I/apps/OPENMPI/4.0.5/GCC/.../libevent2022/libevent/include \
  -I/apps/OPENMPI/4.0.5/GCC/include \
  -pthread -Wl,-rpath -Wl,/apps/OPENMPI/4.0.5/GCC/lib \
  -Wl,--enable-new-dtags -L/apps/OPENMPI/4.0.5/GCC/lib -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-AMD machine, the following FORTRAN compilers (along others) are available:

- flang: This is an AMD-optimized compiler for FORTRAN (from AOCC). Help can be obtained from:

  ```
  $ man flang
  ```

- 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 ouput for

```
$ mpif90 -show
```

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
```

Shared-memory parallelism with FORTRAN

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

```
$ mpif77 -fopenmp -o exename filename.f
$ mpif90 -fopenmp -o exename filename.f
```

5.3.4 Batch system on the CTE-AMD system

Like the DEEP system, see Sec. 2.3.4, the CTE-AMD system uses the SLURM scheduler for batch processing support. This section provides information for getting started with job execution on the CTE-AMD system. Further details on SLURM's environment variables and how to interpret job status and reason codes can be found in Annex A Additional information on SLURM.

Submitting jobs

Job submission makes use of the `sbatch` command. The `squeue` command shows the status of the jobs, and with `scancel` jobs can be removed from the execution list. Corresponding examples can be found in Sec. 2.3.4.

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. 5.3.1. 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

In addition to the explanations given in Sec. 2.3.4, the following job directives may be helpful on the CTE-AMD system.

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

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

The pathname can be specified by the directive:

```
#SBATCH -D pathname
```

Similar to the DEEP system, GPGPUs can be requested as an additional source. Here the maximum number of GPGPUs is 2:

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

On the CTE-AMD 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
```

A job can be defined to use `X11` as a graphical interface, i.e., the user will 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` will assign the necessary resources to the job and will set 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 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
```

SBATCH examples

Job examples can be found in Sec. 2.3.4. It should be noted that the job file directives need to be adapted to the CTE-AMD system and that the partition names are different to the DEEP system.

Annex A Additional information on SLURM

The following two sub-annexes provide additional information on SLURM's environment variables as well as job status and reason codes. Further information on SLURM can be obtained from SLURM's website[20].

A.1 SLURM environment variables

The SLURM controller will set new variables in the environment of the job script. The following Table 9 lists some of the most relevant environment variables.

Variable	Meaning
SLURM_JOBID	Specifies the job ID of the executing job.
SLURM_NPROCS	Specifies the total number of processes in the job.
SLURM_NNODES	The actual number of nodes assigned to run your 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 9: SLURM environment variables.

A.2 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, they will be followed by a reason. SLURM uses codes to display this information. The most relevant codes are displayed in the following Table 10.

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.

[20] SLURM website https://slurm.schedmd.com

Code	Meaning
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 10: Status codes of SLURM.

The list in Table 11 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. Your job will eventually run.
Dependency	This job is waiting for a dependent job to complete and will run afterwards.
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; job will run eventually.
QOSGrpMaxJobsLimit	Maximum number of jobs for your job's QoS have been met; job will run eventually.
QOSGrpNodeLimit	All nodes assigned to your job's specified QoS are in use; job will run eventually.
PartitionCpuLimit	All CPUs assigned to your job's specified partition are in use; job will run eventually.
PartitionMaxJobsLimit	Maximum number of jobs for your job's partition have been met; job will run eventually.
PartitionNodeLimit	All nodes assigned to your job's specified partition are in use; job will run eventually.
AssociationCpuLimit	All CPUs assigned to your job's specified association are in use; job will run eventually.

Code	Meaning
`AssociationMaxJobsLimit`	All CPUs assigned to your job's specified association are in use; job will run eventually.
`AssociationNodeLimit`	All nodes assigned to your job's specified association are in use; job will run eventually.

Table 11: SLURM reason codes.

List of Acronyms and Abbreviations

AA	Active Archive
AI	Artificial intelligence
BSC	Barcelona Supercomputing Center
CIDR	Classless Inter-Domain Routing
CoE RAISE	European Center of Excellence in Exascale Computing “Research on AI- and Simulation-Based Engineering at Exascale”
CM	Cluster module
CN	Cluster node
DAM	Data analytics module
DCPMM	DC Persistent Memory Module
DEEP	Dynamical Exascale Entry Platform
DEEP-EST	DEEP - Extreme Scale Technologies
DPC++	Data Parallel C++
DRAM	Dynamic random-access memory
ESB	Extreme-Scale Booster
EXT4	Extended File System (v4)
FEFS	Fujitsu Exabyte File System
FPGA	Field Programmable Gate Array
FZJ	Forschungszentrum Jülich GmbH
GCS	Gauss Centre for Supercomputing
GPA	General-purpose architecture
GPFS	General Parallel File System
GPGPU	General-purpose graphics processing unit
HLRS	High-Performance Center Stuttgart
HPC	High-performance computing
HPDA	High-performance data analytics
JARA-HPC	Jülich Aachen Research Alliance High-Performance Computing
JURECA	Jülich Research on Exascale Cluster Architectures
JUWELS	Jülich Wizard for European Leadership Science
JSC	Jülich Supercomputing Centre
JUST	Jülich Storage Cluster
KNL	Knight’s Landing
LRZ	Leibnitz Supercomputing Centre of the Bavarian Academy and Sciences and Humanities
MCDRAM	Multi-channel DRAM
MSA	Modular supercomputing architecture
NFS	Network File System protocol
NIC	John von Neumann-Institute for Computing
NVMe	Non-volatile memory express
OpenBLAS	Open Basic Linear Algebra Subprograms
OpenCL	Open Computing Language
PA	Project administrator
PI	Principal investigator
PRACE	Partnership for Advanced Computing in Europe
PU	Public
RDMA	Random direct memory access
RoCE	RDMA over Converged Ethernet

QoS	Quality of service
PMT	Project management team
RAISE	see CoE RAISE
RTU	Riga Technical University
RWTH	Rheinisch-Westfälische Technische Hochschule Aachen
SDV	Software development vehicle
SIMD	Single instruction, multiple data
SSD	Solid-state disc
SVE	Scalable Vector Extension
TofuD	Tofu Interconnect D network
UOI	University of Iceland
XFS	Extended File System

References

[1] DEEPTRAC; DEEP-EST Wiki https://deeptrac.zam.kfa-juelich.de:8443/trac/wiki/Public/User_Guide

[2] Ajima, Y., Kawashima, T., Okamoto, T., Shida, N., Hirai, K., Shimizu, T., Hiramoto, S., Ikeda, Y., Yoshikawa, T., Uchida, K., and Inoue, T. (2018). The Tofu Interconnect D. 2018 IEEE International Conference on Cluster Computing (CLUSTER), 646–654. https://doi.org/10.1109/CLUSTER.2018.00090