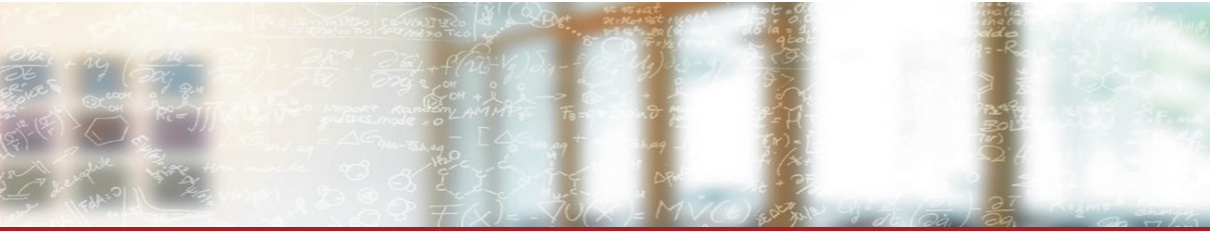
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Introduction to the Piz Daint environment

CSCS-USI Summer School 2019

Vasileios Karakasis, CSCS

July 15, 2019

Overview

- Accessing CSCS
- Compiling my code
- Running my code
- Editing my code
- Transferring files from/to CSCS
- Repository of the course

Piz Daint

Computing nodes

Piz Daint is a hybrid cluster of Cray XC40/XC50 nodes

- Hybrid nodes (XC50)
  - 5320 total
  - Intel® Xeon® E5-2690 v3 @ 2.60GHz (12 cores, 64GB RAM, Haswell)
  - NVIDIA® Tesla® P100 16GB (Pascal)
- Multicore nodes
  - 1813 nodes
  - Two Intel® Xeon®E5-2695 v4 @ 2.10GHz (2 x 18 cores, 64/128 GB RAM, Broadwell)
- Login nodes
  - 5 total
  - Intel® Xeon®CPU E5-2650 v3 @ 2.30GHz (10 cores, 256 GB RAM, Haswell)
- Aries routing and communications ASIC, and Dragonfly network topology

Piz Daint

Filesystems

- `/scratch`: High performance Lustre filesystem accessible from the computing nodes
  - Environment variable `$SCRATCH` points to it
  - Total capacity: 6.2 PB
  - Must be used for heavy I/O
- `/users`: GPFS filesystem for the users' homes
- `/project`, `/store`: Long-term storage for computational projects

More on `https://user.cscs.ch/storage/file_systems/`

Accessing Piz Daint

- Accessible through SSH
- Piz Daint is not directly accessible from the outside world:
  - `ela` → `daint10x` → `nidxxxxx`

Two-steps process:

1. Login to the frontend, forwarding X11 (will be needed the second day)
2. Move to the login nodes of Piz Daint

```
# Login to the frontend first
ssh -Y courseXX@ela.cscs.ch
ssh -Y daint
```

Programming Environments

Cray Linux Programming Environment

- 4 compilers available: `CCE`, `GNU`, `INTEL`, `PGI`
- 4 predefined Programming Environments:
  - PrgEnv-`cray` (default), `PrgEnv-gnu`, `PrgEnv-intel`, `PrgEnv-pgi`
  - `echo $PE_ENV` to get the current PrgEnv
- 3 wrappers available: `ftn` (Fortran), `cc` (C), `CC` (C++)
  - Required for compiling MPI programs
  - They set appropriate optimisation flags for the target architecture (CPU or GPU)
  - They provide a sort of portability across the programming environments

Managing programming environments

Daint uses *Environment Modules* (TMod) for managing the programming environments and the software packages:

- Dynamic modification of a user's environment via *modulefiles*.
- All programming environments and software on Daint is available through modules.
- The compiler wrappers will detect the loaded programming environment and automatically set the correct flags and libraries.

Managing programming environments

Listing modules

- `module list`

```
3. ↔ karakasv@ela1:~ [Default]
stud33@daint105:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.6                                        12) job/2.2.3-6.0.7.1_5.43__g6c4e934.ari
  2) cce/8.7.3                                               13) dvs/2.7_2.2.118-6.0.7.1_10.2__g58b37a2
  3) craype-network-aries                                    14) alps/6.6.43-6.0.7.1_5.45__ga796da32.ari
  4) craype/2.5.15                                           15) rca/2.2.18-6.0.7.1_5.47__g2aa4f39.ari
  5) cray-libsci/18.07.1                                     16) atp/2.1.2
  6) udreg/2.3.2-6.0.7.1_5.13__g5196236.ari                  17) perftools-base/7.0.3
  7) ugni/6.0.14.0-6.0.7.1_3.13__gea11d3d.ari                18) PrgEnv-cray/6.0.4
  8) pmi/5.0.14                                              19) cray-mpich/7.7.2
  9) dmapp/7.1.1-6.0.7.1_5.45__g5a674e0.ari                  20) slurm/17.11.12.cscs-1
 10) gni-headers/5.0.12.0-6.0.7.1_3.11__g3b1768f.ari         21) craype-haswell
 11) xpmem/2.2.15-6.0.7.1_5.11__g7549d06.ari                 22) xalt/daint-2016.11
stud33@daint105:~>
```

Managing programming environments

Switching programming environments

- Switch to the PGI programming environment
- `module switch`

3. karakasv@ela1:~ [Default]

```
stud33@daint107:~> module switch PrgEnv-cray/6.0.4 PrgEnv-pgi
stud33@daint107:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.6                                         12) dmapp/7.1.1-6.0.7.1_5.45__g5a674e0.ari
  2) pgi/18.5.0                                               13) gni-headers/5.0.12.0-6.0.7.1_3.11__g3b1768f.ari
  3) craype-haswell                                           14) xpmem/2.2.15-6.0.7.1_5.11__g7549d06.ari
  4) craype-network-aries                                     15) job/2.2.3-6.0.7.1_5.43__g6c4e934.ari
  5) craype/2.5.15                                            16) dvs/2.7_2.2.118-6.0.7.1_10.2__g58b37a2
  6) cray-mpich/7.7.2                                         17) alps/6.6.43-6.0.7.1_5.45__ga796da32.ari
  7) slurm/17.11.12.cscs-1                                    18) rca/2.2.18-6.0.7.1_5.47__g2aa4f39.ari
  8) xalt/daint-2016.11                                       19) atp/2.1.2
  9) udreg/2.3.2-6.0.7.1_5.13__g5196236.ari                   20) perftools-base/7.0.3
 10) ugni/6.0.14.0-6.0.7.1_3.13__gea11d3d.ari                 21) PrgEnv-pgi/6.0.4
 11) pmi/5.0.14
stud33@daint107:~> ftn -V

pgf90 18.5-0 64-bit target on x86-64 Linux -tp haswell-64
PGI Compilers and Tools
Copyright (c) 2018, NVIDIA CORPORATION.  All rights reserved.
stud33@daint107:~>
```

Managing programming environments

Switching back to the Cray programming environment

3. karakasv@ela1:~ [Default]

```
stud33@daint107:~> module switch PrgEnv-pgi/6.0.4 PrgEnv-cray
stud33@daint107:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.6                                    12) pmi/5.0.14
  2) slurm/17.11.12.cscs-1                               13) dmapp/7.1.1-6.0.7.1_5.45__g5a674e0.ari
  3) xalt/daint-2016.11                                  14) gni-headers/5.0.12.0-6.0.7.1_3.11__g3b1768f.ari
  4) cce/8.7.3                                           15) xpmem/2.2.15-6.0.7.1_5.11__g7549d06.ari
  5) craype-haswell                                      16) job/2.2.3-6.0.7.1_5.43__g6c4e934.ari
  6) craype-network-aries                                17) dvs/2.7_2.2.118-6.0.7.1_10.2__g58b37a2
  7) craype/2.5.15                                       18) alps/6.6.43-6.0.7.1_5.45__ga796da32.ari
  8) cray-mpich/7.7.2                                    19) rca/2.2.18-6.0.7.1_5.47__g2aa4f39.ari
  9) cray-libsci/18.07.1                                 20) atp/2.1.2
 10) udreg/2.3.2-6.0.7.1_5.13__g5196236.ari              21) perftools-base/7.0.3
 11) ugni/6.0.14.0-6.0.7.1_3.13__gea11d3d.ari            22) PrgEnv-cray/6.0.4
stud33@daint107:~> ftn -V
Cray Fortran : Version 8.7.3  Wed Jul 10, 2019  17:47:52
stud33@daint107:~>
```

Managing programming environments

Loading and unloading modules

- `module load [MODULE_NAME]`
- `module unload [MODULE_NAME]`

3. ↣ karakasv@ela1:~ [Default]

```
stud33@daint107:~> module load cray-hdf5
stud33@daint107:~> which h5dump
/opt/cray/pe/hdf5/1.10.2.0/bin/h5dump
stud33@daint107:~> module unload cray-hdf5
stud33@daint107:~> which h5dump
which: no h5dump in (/opt/cray/pe/perftools/7.0.3/bin:/opt/cray/pe/papi/5.6.0.3/bin:/opt/cray/rca/2.2.18-6.0.7.1_5.47__g2aa4f3
9.ari/bin:/opt/cray/alps/6.6.43-6.0.7.1_5.45__ga796da32.ari/sbin:/opt/cray/job/2.2.3-6.0.7.1_5.43__g6c4e934.ari/bin:/opt/cray/
pe/mpt/7.7.2/gni/bin:/opt/cray/pe/craype/2.5.15/bin:/opt/cray/pe/cce/8.7.3/binutils/x86_64/x86_64-pc-linux-gnu/bin:/opt/cray/p
e/cce/8.7.3/binutils/cross/x86_64-aarch64/aarch64-linux-gnu/../bin:/opt/cray/pe/cce/8.7.3/utils/x86_64/bin:/apps/daint/UES/xal
t/0.7.6/bin:/opt/slurm/17.11.12.cscs/bin:/opt/cray/pe/modules/3.2.10.6/bin:/opt/slurm/default/bin:/apps/daint/system/bin:/apps
/common/system/bin:/users/stud33/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/lib/mit/bin:/usr/lib/mit/sbin:/opt/cray/pe
/bin)
stud33@daint107:~> h5dump
If 'h5dump' is not a typo you can use command-not-found to lookup the package that contains it, like this:
    cnf h5dump
stud33@daint107:~>
```

Managing programming environments

Checking available modules

- The `daint-gpu` makes available the CSCS software stack built for the hybrid nodes of the system

```
3. ↣ karakasv@ela1:~ [Default]
stud33@daint107:~> module load daint-gpu
stud33@daint107:~> module avail

----------------------------- /apps/daint/UES/jenkins/6.0.UP07/gpu/easybuild/tools/modules/all -----------------------------
CubeGUI/4.4(default)                                callgraphplugin/0.1(default)
CubeLib/4.4                                         clusteringsuite/2.6.7
CubeLib/4.4-CrayCCE-18.08                           clusteringsuite/2.6.8(default)
CubeLib/4.4-CrayGNU-18.08(default)                  clusteringsuite/latest
CubeLib/4.4-CrayIntel-18.08                         ddt/18.2.1-Suse-12(default)
CubeLib/4.4-CrayPGI-18.08                           ddt/18.3-Suse-12
CubeW/4.4-CrayCCE-18.08                             ddt/19.0-Suse-12
CubeW/4.4-CrayGNU-18.08(default)                    folding/1.3.2(default)
CubeW/4.4-CrayIntel-18.08                           folding/latest
CubeW/4.4-CrayPGI-18.08                             go/1.12.1.linux-amd64(default)
Dimemas/5.3.6(default)                              gperftools/2.7(default)
Dimemas/latest                                      gprof2dot/2017.9.19-CrayGNU-18.08-python3(default)
Extrae/3.5.4-CrayGNU-18.08(default)                 graphviz/2.40.1(default)
IPM/2.0.6-CrayCCE-18.08                             inspector/2018(default)
IPM/2.0.6-CrayGNU-18.08(default)                    inspector/2019
IPM/2.0.6-CrayIntel-18.08                           inspector/2019_update4
IPM/2.0.6-CrayPGI-18.08                             jengafettplugin/0.1(default)
MUST/v1.6-rc1-CrayGNU-18.08(default)                libelf/0.8.13(default)
Paraver/4.7.2(default)                              likwid/4.3.3-perf_event(default)
```

Managing programming environments

Checking available modules

- Check available versions of a software

```
3. ↣ karakasv@ela1:~ [Default]
stud33@daint107:~> module avail gcc

------------------------------------------------------- /opt/modulefiles -------------------------------------------------------
gcc/4.9.3           gcc/6.1.0           gcc/7.1.0           gcc/8.1.0
gcc/5.3.0           gcc/6.2.0(default)  gcc/7.3.0           gcc/8.3.0
stud33@daint107:~>
```

Managing programming environments

Get information about a module

- Environment variables set, paths etc.

3. ↣ karakasv@ela1:~ [Default]

```
stud33@daint107:~> module show gcc
-------------------------------------------------------------------
/opt/modulefiles/gcc/6.2.0:

conflict         gcc
conflict         gcc-cross-aarch64
prepend-path     PATH /opt/gcc/6.2.0/bin
prepend-path     MANPATH /opt/gcc/6.2.0/snos/share/man
prepend-path     INFOPATH /opt/gcc/6.2.0/snos/share/info
prepend-path     LD_LIBRARY_PATH /opt/gcc/6.2.0/snos/lib64
setenv           GCC_PATH /opt/gcc/6.2.0
setenv           GCC_VERSION 6.2.0
setenv           GNU_VERSION 6.2.0
-------------------------------------------------------------------

stud33@daint107:~>
```

Managing programming environments

Get help for a module

3. karakasv@ela1:~ [Default]

```
stud33@daint107:~> module help CP2K

----------- Module Specific Help for 'CP2K/6.1-CrayGNU-18.08-cuda-9.1' ---------------------------



Description
===========
CP2K is a freely available (GPL) program, written in Fortran 95, to perform atomistic and molecular
 simulations of solid state, liquid, molecular and biological systems. It provides a general framework for different
 methods such as e.g. density functional theory (DFT) using a mixed Gaussian and plane waves approach (GPW), and
 classical pair and many-body potentials.


More information
================
 - Homepage: http://www.cp2k.org/

stud33@daint107:~>
```

Running on Piz Daint

The job scheduler

Piz Daint uses native SLURM for running jobs on the compute nodes. There are three ways of submitting a job:

1. Interactively from the login nodes using the `srun` command.
2. By submitting a job script using the `sbatch` command.
3. By pre-allocating resources using the `salloc` command.

Running on Piz Daint

Using the `srun` command

Necessary and useful options:

- `-C gpu`: requests allocation on the hybrid (GPU) nodes (required)
- `--reservation=summer_school`
  - 64 nodes valid until Jul. 25 @ 13:00.
- `-N 2`: number of compute nodes (default is 1)
- `-n 2`: number of MPI tasks (default is 1)
- `-t 5`: maximum duration of the job (default is 30min)
  - Allows to get an allocation quicker
  - Job will be killed if time limit is reached
  - Maximum time slot for a job is 24h

More on `https://user.cscs.ch/access/running`

Running on Piz Daint

Using the `srun` command

3. karakasv@ela1:~ [Default]

```
stud33@daint107:~> srun --reservation=summer_school_test -Cgpu -t1 -N2 hostname
srun: job 15391913 queued and waiting for resources
srun: job 15391913 has been allocated resources
nid02296
nid02822
stud33@daint107:~> srun --reservation=summer_school_test -Cgpu -t1 -n2 hostname
srun: job 15391932 queued and waiting for resources
srun: job 15391932 has been allocated resources
nid02296
nid02296
stud33@daint107:~>
```

Running on Piz Daint

Using the `sbatch` command

```
3. ↣ karakasv@ela1:~ [Default]
stud33@daint107:~> cat job.sh
#!/bin/bash
#SBATCH -J 'my_first_job'
#SBATCH -C gpu
#SBATCH -N 2
#SBATCH --reservation=summer_school_test
#SBATCH -o myjob.out
#SBATCH -e myjob.err

echo "My job ID is $SLURM_JOB_ID"
srun hostname
stud33@daint107:~> sbatch job.sh
Submitted batch job 15391950
stud33@daint107:~> squeue -j 15391950
   JOBID     USER ACCOUNT           NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES CPUS
15391950   stud33   std01   my_first_job  CF None      18:43:19                  0:02      29:58     2   48
stud33@daint107:~> squeue -j 15391950
   JOBID     USER ACCOUNT           NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES CPUS
15391950   stud33   std01   my_first_job   R None      18:43:19                  0:09      29:51     2   48
stud33@daint107:~>
```

Running on Piz Daint

Using the `sbatch` command – Examinining the output

3. karakasv@ela1:~ [Default]

```
stud33@daint107:~> cat myjob.err
stud33@daint107:~> cat myjob.out
My job ID is 15391950
nid02296
nid02822

Batch Job Summary Report for Job "my_first_job" (15391950) on daint
----------------------------------------------------------------------------------------------------
             Submit            Eligible               Start                 End    Elapsed  Timelimit
------------------ ------------------ ------------------ ------------------- ---------- ----------
2019-07-10T18:43:08 2019-07-10T18:43:08 2019-07-10T18:43:19 2019-07-10T18:43:34   00:00:15   00:30:00
----------------------------------------------------------------------------------------------------
Username    Account     Partition   NNodes   Energy
----------  ----------  ----------  ------  --------------
stud33      std01       normal           2    1.44K joules

This job did not utilize any GPUs

-------------------------------------------------------------
Scratch File System        Files       Quota
--------------------  ----------  ----------
/scratch/snx3000              33     1000000
```

Running on Piz Daint

Using the `salloc` command

```
3. ↣ karakasv@ela1:~ [Default]
stud33@daint107:~> salloc --reservation=summer_school_test -Cgpu -t1 -N2
salloc: Pending job allocation 15392032
salloc: job 15392032 queued and waiting for resources
salloc: job 15392032 has been allocated resources
salloc: Granted job allocation 15392032
stud33@daint107:~> srun -N2 hostname
nid02296
nid02822
stud33@daint107:~> srun -N1 hostname
nid02296
stud33@daint107:~> srun -N4 hostname
srun: error: Only allocated 2 nodes asked for 4
stud33@daint107:~> exit
salloc: Relinquishing job allocation 15392032
stud33@daint107:~>
```

Running on Piz Daint

Other useful commands

- `squeue [OPTIONS]`: Check the status of the system job queue
  - Useful options: `-u [USERNAME]`, `-j [JOBID]`
- `scancel [JOBID]`: Cancel a job
- `scontrol`: Detailed information about partitions, reservations, computing nodes etc.

Running on Piz Daint

Other useful commands

3. karakasv@ela1:~ [Default]

```
stud33@daint107:~> scontrol show reservation summer_school
ReservationName=summer_school StartTime=Mon 12:00 EndTime=25 Jul 13:00 Duration=10-01:00:00
   Nodes=nid0[6144-6207] NodeCnt=64 CoreCnt=768 Features=gpu PartitionName=normal Flags=
   TRES=cpu=1536
   Users=(null) Accounts=std01,u3 Licenses=(null) State=INACTIVE BurstBuffer=(null) Watts=n/a

stud33@daint107:~> scontrol show partition normal
PartitionName=normal
   AllowGroups=ALL AllowAccounts=ALL AllowQos=ALL
   AllocNodes=ALL Default=YES QoS=N/A
   DefaultTime=00:30:00 DisableRootJobs=NO ExclusiveUser=NO GraceTime=0 Hidden=NO
   MaxNodes=2400 MaxTime=1-00:00:00 MinNodes=1 LLN=NO MaxCPUsPerNode=UNLIMITED
   Nodes=nid[00004-00007,00012-00024,00026-00062,00064-00067,00072-00126,00128-00190,00192-00195,00200-00254,00260-00318,00320
-00323,00328-00382,00388-00446,00456-00510,00516-00574,00576-00579,00584-00638,00644-00702,00704-00707,00712-00766,00772-00830
,00832-00835,00840-00894,00900-00958,00960-00963,00968-01022,01028-01086,01088-01150,01152-01192,01194-01214,01216-01278,01280
-01723,01928-01935,01940-01967,01972-02319,02324-02351,02356-02703,02708-02735,02740-03087,03092-03119,03124-03471,03476-03503
,03512-03855,03860-03887,03892-04239,04244-04271,04280-07679]
   PriorityJobFactor=10 PriorityTier=20 RootOnly=NO ReqResv=NO OverSubscribe=EXCLUSIVE
   OverTimeLimit=NONE PreemptMode=OFF
   State=UP TotalCPUs=253872 TotalNodes=7318 SelectTypeParameters=NONE
   DefMemPerNode=UNLIMITED MaxMemPerNode=UNLIMITED

stud33@daint107:~>
```

Editing files

- `vim` or `gvim` (X version)
- `emacs -nw` or just `emacs` (X version)
- `gedit` (X only)

Moving data to/from CSCS

- `scp`: Remote copy over SSH
  - Getting a file: `scp studXX@ela.cscs.ch:remotefile localfile`
  - Getting a directory: `scp -r studXX@ela.cscs.ch:remotedir localdir`
  - Sending a file: `scp localfile studXX@ela.cscs.ch:remotefile`
  - Sending a directory: `scp localdir studXX@ela.cscs.ch:remotedir`
- `rsync`: Synchronize files remotely over SSH
  - `rsync -avz studXX@ela.cscs.ch:remotedir/ localdir/`
  - `rsync -avz localdir/ studXX@ela.cscs.ch:remotedir/`
  - Pay attention to the slashes! `rsync` *behaves differently with or without slashes.*

Summer school repository

All the material of the course is placed inside the following Github repo:

- `https://github.com/eth-cscs/SummerSchool2019`
- For instructions on how to clone and pull from the repository, check its front page.

Organization of the repository

- `miniapp/`: The different versions of the mini-app that you will work throughout the summer school + slides.
- `topics/`: The practical exercises of the different topics that will be covered during the the summer school + slides.
- `scripts/`: Useful scripts for the exercises and the mini-app.

Solutions:

- The solutions of the exercises and the mini-app will appear at the end of the summer school in subfolders named `solutions/` under each respective topic.