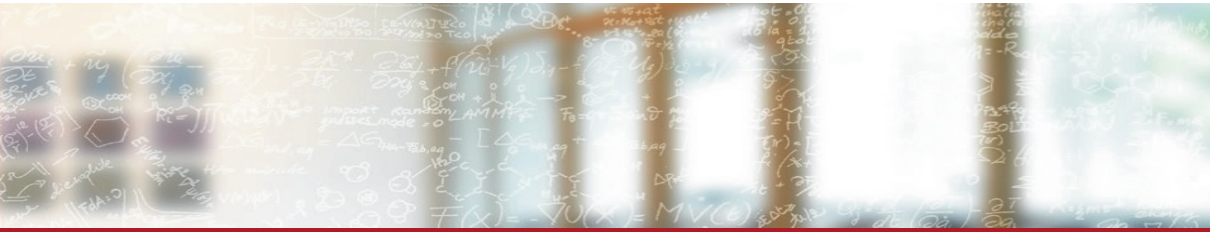
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Introduction to the Piz Daint environment

CSCS-USI Summer School 2021

Vasileios Karakasis, CSCS

July 19, 2021

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: 8.8 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 classXXX@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
Currently Loaded Modulefiles:
  1) modules/3.2.11.3                                  12) job/2.2.4-7.0.1.1_3.8__g36b56f4.ari
  2) cce/9.0.2                                         13) dvs/2.12_2.2.151-7.0.1.1_5.6__g7eb5e703
  3) craype-network-aries                              14) alps/6.6.56-7.0.1.1_4.10__g2e60a7e4.ari
  4) craype/2.6.1                                      15) rca/2.2.20-7.0.1.1_4.9__g8e3fb5b.ari
  5) cray-libsci/19.06.1                               16) atp/2.1.3
  6) udreg/2.3.2-7.0.1.1_3.9__g8175d3d.ari             17) perftools-base/7.1.1
  7) ugni/6.0.14.0-7.0.1.1_7.10__ge78e5b0.ari          18) PrgEnv-cray/6.0.5
  8) pmi/5.0.14                                        19) cray-mpich/7.7.10
  9) dmapp/7.1.1-7.0.1.1_4.8__g38cf134.ari             20) slurm/20.02.2-1
 10) gni-headers/5.0.12.0-7.0.1.1_6.7__g3b1768f.ari    21) craype-haswell
 11) xpmem/2.2.19-7.0.1.1_3.7__gdcf436c.ari            22) xalt/2.7.24
```

Managing programming environments

Switching programming environments

```
$ module switch PrgEnv-cray/6.0.5 PrgEnv-pgi
$ module list
Currently Loaded Modulefiles:
  1) modules/3.2.11.3                          12) dmapp/7.1.1-7.0.1.1_4.8__g38cf134.ari
  2) pgi/19.7.0                                13) gni-headers/5.0.12.0-7.0.1.1_6.7__g3b1768f.ari
  3) craype-haswell                            14) xpmem/2.2.19-7.0.1.1_3.7__gdcf436c.ari
  4) craype-network-aries                      15) job/2.2.4-7.0.1.1_3.8__g36b56f4.ari
  5) craype/2.6.1                              16) dvs/2.12_2.2.151-7.0.1.1_5.6__g7eb5e703
  6) cray-mpich/7.7.10                         17) alps/6.6.56-7.0.1.1_4.10__g2e60a7e4.ari
  7) slurm/20.02.2-1                           18) rca/2.2.20-7.0.1.1_4.9__g8e3fb5b.ari
  8) xalt/2.7.24                               19) atp/2.1.3
  9) udreg/2.3.2-7.0.1.1_3.9__g8175d3d.ari     20) perftools-base/7.1.1
 10) ugni/6.0.14.0-7.0.1.1_7.10__ge78e5b0.ari  21) PrgEnv-pgi/6.0.5
 11) pmi/5.0.14
```

Managing programming environments

Switching back to the Cray programming environment

```
$ module switch PrgEnv-pgi/6.0.5 PrgEnv-cray
$ module list
Currently Loaded Modulefiles:
  1) modules/3.2.11.3                          12) pmi/5.0.14
  2) slurm/20.02.2-1                           13) dmapp/7.1.1-7.0.1.1_4.8__g38cf134.ari
  3) xalt/2.7.24                               14) gni-headers/5.0.12.0-7.0.1.1_6.7__g3b1768f.ari
  4) cce/9.0.2                                 15) xpmem/2.2.19-7.0.1.1_3.7__gdcf436c.ari
  5) craype-haswell                            16) job/2.2.4-7.0.1.1_3.8__g36b56f4.ari
  6) craype-network-aries                      17) dvs/2.12_2.2.151-7.0.1.1_5.6__g7eb5e703
  7) craype/2.6.1                              18) alps/6.6.56-7.0.1.1_4.10__g2e60a7e4.ari
  8) cray-mpich/7.7.10                         19) rca/2.2.20-7.0.1.1_4.9__g8e3fb5b.ari
  9) cray-libsci/19.06.1                       20) atp/2.1.3
 10) udreg/2.3.2-7.0.1.1_3.9__g8175d3d.ari     21) perftools-base/7.1.1
 11) ugni/6.0.14.0-7.0.1.1_7.10__ge78e5b0.ari  22) PrgEnv-cray/6.0.5
```

Managing programming environments

Loading and unloading modules

```
$ module load cray-hdf5
$ which h5dump
/opt/cray/pe/hdf5/1.10.5.1/bin/h5dump
$ module unload cray-hdf5
$ which h5dump
which: no h5dump in (/opt/cray/pe/perftools/7.1.1/bin:/opt/cray/pe/papi/5.7.0.2/bin:/opt/cray/rca
     ➥/2.2.20-7.0.1.1_4.9__g8e3fb5b.ari/bin:/opt/cray/alps/6.6.56-7.0.1.1_4.10__g2e60a7e4.ari/sbin:/opt
     ➥/cray/alps/default/bin:/opt/cray/job/2.2.4-7.0.1.1_3.8__g36b56f4.ari/bin:/opt/cray/pe/mpt/7.7.10/
     ➥gni/bin:/opt/cray/pe/craype/2.6.1/bin:/opt/cray/pe/cce/9.0.2/cce-clang/x86_64/bin:/opt/cray/pe/
     ➥cce/9.0.2/binutils/x86_64/x86_64-pc-linux-gnu/bin:/opt/cray/pe/cce/9.0.2/binutils/cross/x86_64-
     ➥aarch64/aarch64-linux-gnu/../bin:/opt/cray/pe/cce/9.0.2/utils/x86_64/bin:/users/karakasv/local/
     ➥bin:/users/karakasv/local/sbin:/apps/daint/UES/xalt/xalt2/software/xalt/2.7.24/sbin:/apps/daint/
     ➥UES/xalt/xalt2/software/xalt/2.7.24/bin:/opt/cray/pe/modules/3.2.11.3/bin:/apps/daint/system/bin
     ➥:/apps/common/system/bin:/usr/local/bin:/usr/bin:/bin:/usr/lib/mit/bin:/usr/lib/mit/sbin:/opt/
     ➥cray/pe/bin)
$ h5dump
If 'h5dump' is not a typo you can use command-not-found to lookup the package that contains it, like
     ➥this:
    cnf h5dump
```

Managing programming environments

Checking available modules

```
$ module avail gcc

------------------------------------- /opt/modulefiles ---------------------------------------
gcc/6.1.0          gcc/7.3.0          gcc/8.1.0          gcc/8.2.0          gcc/8.3.0(default)
```

Managing programming environments

Loading the CSCS software stack

```
$ module load daint-gpu
$ module avail TensorFlow

---------------- /apps/daint/UES/jenkins/7.0.UP01/gpu/easybuild/modules/all -----------------
TensorFlow/1.14.0-CrayGNU-19.10-cuda-10.1.168-python3
TensorFlow/2.0.0-CrayGNU-19.10-cuda-10.1.168
TensorFlow/2.2.0-CrayGNU-19.10-cuda-10.1.168
```

Managing programming environments

Get information about a module

```
$ module show gcc
-------------------------------------------------------------------
/opt/modulefiles/gcc/8.3.0:

conflict         gcc
conflict         gcc-cross-aarch64
prepend-path     PATH /opt/gcc/8.3.0/bin
prepend-path     MANPATH /opt/gcc/8.3.0/snos/share/man
prepend-path     INFOPATH /opt/gcc/8.3.0/snos/share/info
prepend-path     LD_LIBRARY_PATH /opt/gcc/8.3.0/snos/lib64
setenv           GCC_PATH /opt/gcc/8.3.0
setenv           GCC_VERSION 8.3.0
setenv           GNU_VERSION 8.3.0
-------------------------------------------------------------------
```

Managing programming environments

Get help for a module

```
$ module help TensorFlow

----------- Module Specific Help for 'TensorFlow/2.2.0-CrayGNU-19.10-cuda-10.1.168'
     ↪---------------------------


Description
===========
An open-source software library for Machine Intelligence.


More information
================
 - Homepage: https://www.tensorflow.org/


Included extensions
===================
grpcio-1.24.0, tensorboard-2.2.1, tensorflow-estimator-2.2.0
```

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`
  - 210 nodes valid until July 24 @ 20:30.
- `-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

```
$ srun --reservation=summer_school -Cgpu -t1 -N2 hostname
srun: job 24097869 queued and waiting for resources
srun: job 24097869 has been allocated resources
nid01994
nid01995
```

Running on Piz Daint

Using the `sbatch` command

```
$ cat > job.sh
#!/bin/bash
#SBATCH -J 'my_first_job'
#SBATCH -C gpu
#SBATCH -N 2
#SBATCH --reservation=summer_school
#SBATCH -o myjob.out
#SBATCH -e myjob.err

echo "My job id is $SLURM_JOB_ID"
srun hostname
^D
$ sbatch job.sh
Submitted batch job 24101970
$ squeue -j 24101970
   JOBID     USER ACCOUNT           NAME  ST REASON    START_TIME           TIME  TIME_LEFT NODES CPUS
24101970 karakasv csstaff   my_first_job  PD Priority  N/A                  0:00        1:00     2    2
$ squeue -j 24101970
   JOBID     USER ACCOUNT           NAME  ST REASON    START_TIME           TIME  TIME_LEFT NODES CPUS
24101970 karakasv csstaff   my_first_job   R None      18:14:26             0:21        0:39     2   48
$ cat myjob.err
$ cat myjob.out
My job id is 24101970
nid01985
nid01986
```

Running on Piz Daint

Using the `salloc` command

```
$ salloc -Cgpu -t1 -N2
salloc: Pending job allocation 24102049
salloc: job 24102049 queued and waiting for resources
salloc: job 24102049 has been allocated resources
salloc: Granted job allocation 24102049
$ srun -N2 hostname
nid01986
nid01985
$ srun -N1 hostname
nid01985
$ srun -N4 hostname
srun: error: Only allocated 2 nodes asked for 4
$ exit
salloc: Relinquishing job allocation 24102049
salloc: Job allocation 24102049 has been revoked.
```

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

```
$ scontrol show reservation summer_school
ReservationName=summer_school StartTime=Mon 08:30 EndTime=Mon 20:30 Duration=12:00:00
   Nodes=nid0[3860-3887,3892-4073] NodeCnt=210 CoreCnt=2520 Features=(null) PartitionName=(null) Flags=
        ➥WEEKDAY,SPEC_NODES
   TRES=cpu=5040
   Users=(null) Accounts=class02,class03,class04,csstaff Licenses=(null) State=INACTIVE BurstBuffer=(
        ➥null) Watts=n/a
   MaxStartDelay=(null)

[18:22:47] karakasv@daint106 [~]
$ 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=nid0
        ➥[0004-0007,0012-0024,0026-0062,0064-0067,0072-0126,0128-0190,0192-0195,0200-0254,0260-0318,0320-0323,0
        ➥
   PriorityJobFactor=10 PriorityTier=20 RootOnly=NO ReqResv=NO OverSubscribe=EXCLUSIVE
   OverTimeLimit=NONE PreemptMode=OFF
   State=UP TotalCPUs=246720 TotalNodes=7212 SelectTypeParameters=NONE
   JobDefaults=(null)
   DefMemPerNode=UNLIMITED MaxMemPerNode=UNLIMITED
```

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 classXXX@ela.cscs.ch:remotefile localfile`
  - Getting a directory: `scp -r classXXX@ela.cscs.ch:remotedir localdir`
  - Sending a file: `scp localfile classXXX@ela.cscs.ch:remotefile`
  - Sending a directory: `scp localdir classXXX@ela.cscs.ch:remotedir`
  - NOTE: The `$SCRATCH` filesystem is not mounted in the frontend nodes
- `rsync`: Synchronize files remotely over SSH
  - `rsync -avz classXXX@ela.cscs.ch:remotedir/ localdir/`
  - `rsync -avz localdir/ classXXX@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/SummerSchool2021`
- 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.