The Piz Daint supercomputer

High Performance Computing for Science and Engineering II

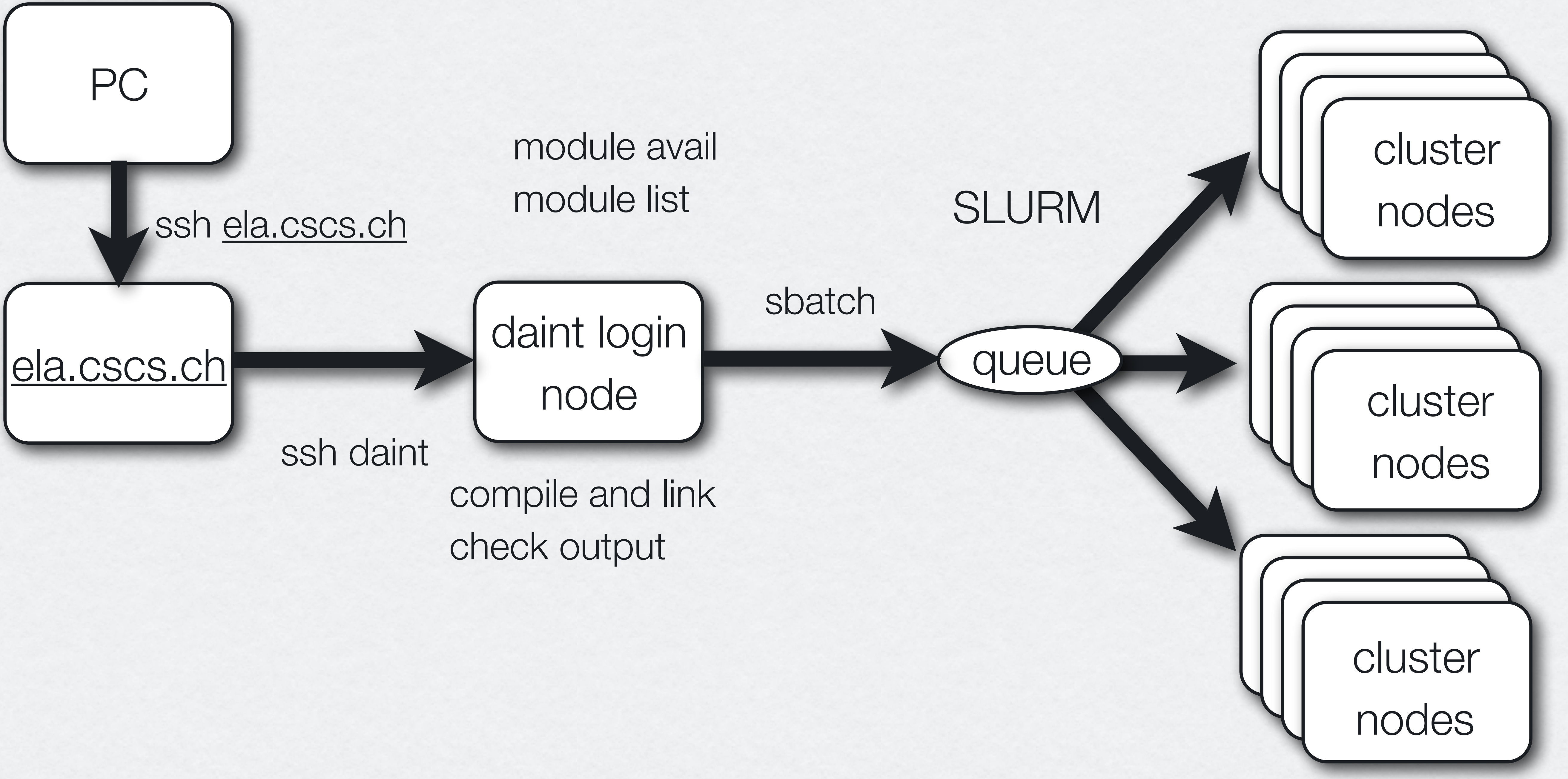
June 8, 2017

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Piz Daint

- http://www.cscs.ch/computers/piz_daint/index.html
- You have access to the Cray XC50 compute nodes
  - Intel Xeon E5-2690 v3 @ 2.60GHz (12 cores, 64GB RAM)
  - NVIDIA Tesla P100 16GB

Accessing and using Pix Daint



PC

ssh ela.cscs.ch

ela.cscs.ch

ssh daint

daint login node

module avail
module list

compile and link
check output

sbatch

queue

SLURM

cluster nodes

cluster nodes

cluster nodes

Basic steps

1. Connect to a login node of Piz Daint
2. Develop your code (copy files or edit)
3. Compile your program
4. Prepare your job script
5. Submit a job / run your program on compute nodes
6. Check your job (status and output)

1. Connect

- ssh studXX@ela.cscs.ch
    - Access to one of the login nodes of Ela (front-end system)
- From ela.cscs.ch: ssh daint
    - Access to one of the login nodes of Piz Daint

2. Develop

- Copy your files to Ela, e.g.
  - scp code.tar.gz <username>@ela.cscs.ch:*code.tar.gz*
- Use a text editor to write/modify your code

3. Compile

- You will need the appropriate programming tools and libraries to compile your code
- Just load the environment module you need
- First of all: **module load daint-gpu**
    - Even if you do not use the GPUs of the compute nodes

3. Compile

- Examples
  - module list (shows loaded modules)
  - module avail (what is available)
  - **module unload PrgEnv-cray** (unload Cray environment)
  - **module load PrgEnv-gnu** (load GNU environment)
  - module load cudatoolkit (load NVIDIA environment)

- The GNU programming environment is highly recommended

3. Compile

- Compile your code and produce the executable
  - cc for C compiler
  - CC for C++ compiler (MPI flags are automatically included)
  - nvcc: CUDA compiler
- Example:
  - cc -O3 -fopenmp hybrid.c -o hybrid

4. Preparing your script

- https://user.cscs.ch/getting_started/running_jobs/jobscript_generator/

```
#!/bin/bash -l
#SBATCH --job-name=test
#SBATCH --time=01:00:00
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
#SBATCH --ntasks-per-core=1
#SBATCH --cpus-per-task=6
#SBATCH --partition=normal
#SBATCH --constraint=gpu

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
export CRAY_CUDA_MPS=1

srun ./hybrid
```

5. Submit your job

- The login nodes are used only for development
- The program must run on a compute node
- To do that, you must use the sbatch command:
  sbatch script

6. Check your job

- Job output messages are written in:
    - slurm-<jobid>.out
- Some useful commands
    - sbuckcheck: displays available computational budget
    - squeue : displays information about jobs
    - squeue -u studXX : displays information about jobs
    - scancel <jobID>: cancels a job

Additional information

- Use the scratch space for running your experiments
  - /scratch/snx3000/$USER

- Visit https://user.cscs.ch, and study (at least) the following:
  - Getting Started: Running Jobs / Piz Daint
  - Getting Started: FAQ
  - Scientific Computing: Code Compilation

- For questions, you **must not** contact the CSCS help
  - Instead use: hpcse_fs17_ta@sympa.ethz.ch