NCCR Cluster User-Guide

University of Basel
M. Devereux

Cluster Specs:

Total Cores:	920
Compute Nodes:	45
Main Storage:	73 TB
Network Communication:	Omni-Path
OS:	Centos 7.4.1708
Job Scheduler:	Slurm 17.11

Node Specs:

CPUs:	2x Intel Xeon E5-2630 v4
Cores:	20
RAM:	32 GB
Scratch space:	1 TB

Table of Contents

1. Cluster Access

1 Overview

Access is possible from within and also from outside the University of Basel network. From outside the university connection is made possible via an intermediate “login node” using an SSH-key. From inside the university network direct connection to the master node using a local user account is possible:

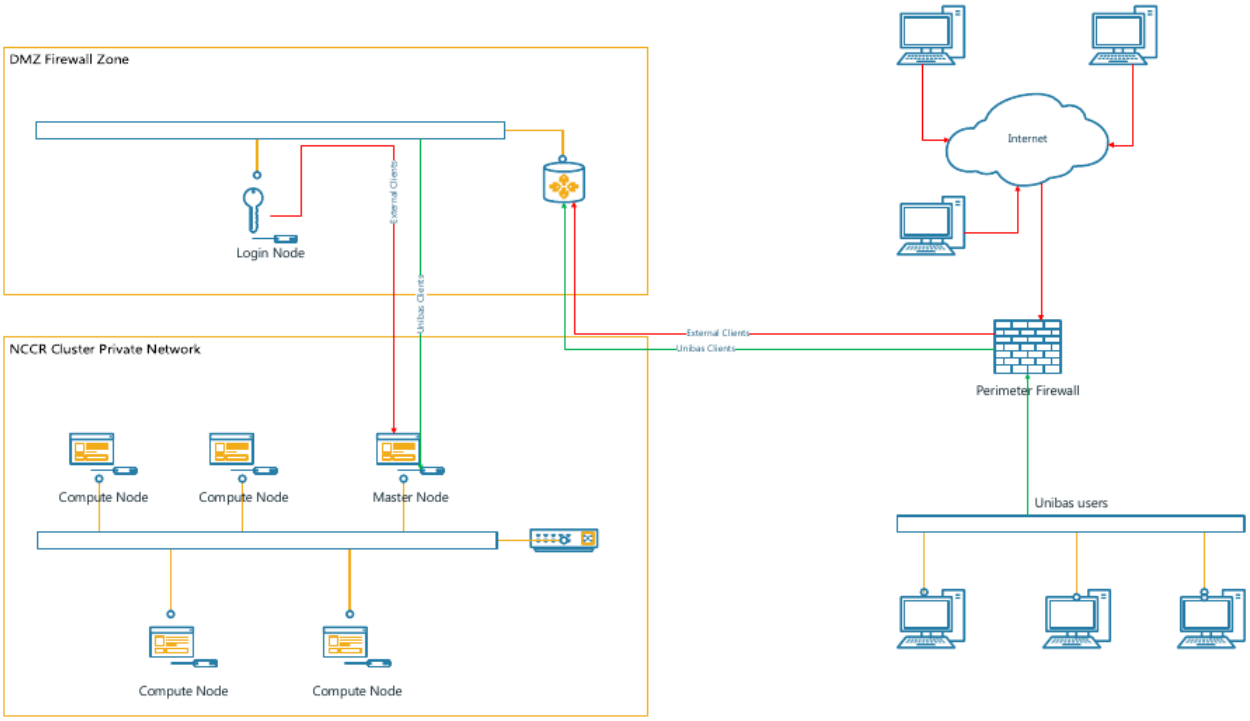

Figure 1: Network schematic for NCCR cluster access

Note that while it is possible to reach the cluster from the university of Basel network, it is not possible to reach machines within the network from the cluster due to firewalls rules of the “DMZ” firewall zone.

2 Getting an Account

Please first contact the cluster administrator (Michael.Devereux@unibas.ch) to obtain an account. You will then be asked to provide a public SSH key for passwordless login to the login node. You should keep the private key safe on your local client machine, readable only by yourself. The procedure you follow to generate the key pair depends on your system, suggested tutorials include:

https://www.ssh.com/ssh/keygen/ (general information)
https://help.ubuntu.com/community/SSH/OpenSSH/Keys (Specific to Ubuntu)

We recommend using at least rsa with a 2048 bit key, preferably a 4096 bit key (ssh-keygen -t rsa -b 4096) or a more secure algorithm such as ecdsa if your local machine supports it (ssh-keygen -t ecdsa -b 521). Please also use a password when you generate the private key so that anyone obtaining your key does not gain automatic access to the cluster.

3 Connecting via SSH

From inside the university you can follow the usual protocol, with the "-Y" flag to launch graphical programs across the network if necessary. Note that you can only connect to the master from your local machine and not from the master to your local machine, so when copying files:

local:~ scp file1 pc-nccr-cluster.chemie.unibas.ch:

will work while:

master:~ scp file1 local-host.chemie.unibas.ch:

will fail to connect due to University firewall rules.

From outside the university you can either firstly connect to the login node, then from there connect to the cluster master node:

local:~ ssh pc-nccr-login.chemie.unibas.ch

pc-nccr-login:~ ssh pc-nccr-cluster

or, more conveniently, you can set up a tunnel through the login node direct to the master node. Assuming your private key is stored in ~/.ssh/ on your local machine, you need to create a file in that folder named "~/.ssh/config" with contents:

Host pc-nccr-cluster.chemie.unibas.ch

ProxyCommand ssh pc-nccr-login.chemie.unibas.ch -W %h:%p -oClearAllForwardings=yes

Now, when you type "ssh pc-nccr-cluster.chemie.unibas.ch" on your local machine you should be directed directly to the master node of the cluster.

Using the Cluster

4 Disk Management

You should store data on /home and /data on the cluster. Both belong to the same 73 TB RAID volume and partition, so the amount of space available is the same. These shares are mounted on every node and NFS communication via Omni-Path ensures improved I/O performance.

Each node additionally has a local 1 TB “scratch” disk (HDD) mounted to /scratch. To avoid placing unnecessary load on the network and slowing down access to /home and /data for other users, you should write all temporary files that will not be kept, and all files that are constantly being rewritten during a run to /scratch. It may be simplest to copy an entire input directory to /scratch at the start of a run, then copy final files back at the end of the run and removing any remaining files. **Note that files in /scratch that are older than ~8 weeks will be automatically deleted.**

5 Slurm Job Scheduler

Slurm is used on many modern clusters to manage job scheduling and has been well documented elsewhere. See for example https://slurm.schedmd.com/quickstart.html

Partitions

Queues, or “partitions”, are easily configurable and adapted to the approximate usage profile of the cluster at a given time. If the current configuration does not suit your needs then please contact an administrator as changes can be made. The current configuration (Dec. 2018) is:

Nodes	Queue	Time-Limit (hr
1-25	short	6
26-45	infinite	-
1-45	vshort	3

The goal is to balance maximizing usage of the cluster (minimum idle nodes) with minimizing wait-time for jobs to start and sharing usage fairly between multiple users. The current setup therefore:

- Uses a “backfill” scheduling system, meaning that a low-priority job will start only if it will not impede a higher priority job from starting
- The priority of a job is decided using a “FAIR_TREE” system, but all users are assigned equal shares so that only a user’s current and recent resource usage affect the priority of their queued jobs. A user that currently or recently ran a large number of jobs on the cluster receives a lower priority for their queued jobs than a user with low or no recent usage.
- To minimize the number of idle nodes, the “vshort” queue is available on every node so that users can easily submit large numbers of short jobs to the whole cluster. As these jobs cannot run for > 3 hours, they will not block new jobs from starting on the cluster for longer than this time.

- To ensure that the wait-time for new jobs to start is not too long, users can only submit long jobs (> 6 hours) to nodes 26-45 in the “infinite” queue, meaning at least nodes 1-25 will never contain jobs that require > 6 hours to complete.
- The “short” queue increases usage of nodes 1-25 by also opening them up to longer jobs (up to 6 hours), without blocking longer jobs in the infinite queue on nodes 26-45 from starting.

Sample sbatch Scripts

Example 1: Single-core job run from local /scratch disk:

```
#!/bin/bash
#SBATCH -J job_1              # Job name
#SBATCH -o job_1.o%j          # File to redirect STDOUT (%j = job_id)
#SBATCH -e job_1.e%j          # File to redirect STDERR
#SBATCH --nodes=1             # Number of nodes for job
#SBATCH --cpus-per-task=1     # 1 core per task
#SBATCH --ntasks=1            # 1 task (1 single-core job)
#SBATCH --partition=infinite  # Partition (queue) to submit to

sdir=/scratch/$USER/$SLURM_JOB_ID    # define scratch folder on local disk
mkdir -p $sdir                       # create scratch folder
cp -r $SLURM_SUBMIT_DIR/* $sdir/     # copy all files from cwd to scratch folder
cd $sdir

./myjob -i myfile.inp -o myfile.out         # run job from scratch directory

cp *.out $SLURM_SUBMIT_DIR/                 # copy output back to cwd
cd $SLURM_SUBMIT_DIR
rm -r $tdir                                 # clean up scratch files
```

Example 2: OpenMPI job run from local /scratch disk on a single node, note that “-np” flag for mpirun is automatically taken from Slurm “ntasks” parameter:

```
#!/bin/bash

#SBATCH -J job_1              # Job name
#SBATCH -o job_1.o%j          # File to redirect STDOUT (%j = job_id)
#SBATCH -e job_1.e%j          # File to redirect STDERR
#SBATCH --nodes=1             # Number of nodes for job
#SBATCH --cpus-per-task=1     # 1 core per task
#SBATCH --ntasks=8            # Each OMPI thread is seen as a single task (so 1 core per thread)
#SBATCH --partition=infinite  # Partition (queue) to submit to

module load gcc/openmpi-1.10.4-hfi

sdir=/scratch/$USER/$SLURM_JOB_ID    # define scratch folder on local disk
mkdir -p $sdir                       # create scratch folder
cp -r $SLURM_SUBMIT_DIR/* $sdir/     # copy all files from cwd to scratch folder
cd $sdir

mpirun ./myjob -i myfile.inp -o myfile.out          # run OMPI job from scratch directory

cp *.out $SLURM_SUBMIT_DIR/                 # copy output back to cwd
cd $SLURM_SUBMIT_DIR
rm -r $tdir                                 # clean up scratch files
```

Example 3: OpenMPI multinode job

```
#!/bin/bash

#SBATCH -J job_1              # Job name
#SBATCH -o job_1.o%j          # File to redirect STDOUT (%j = job_id)
#SBATCH -e job_1.e%j          # File to redirect STDERR
#SBATCH --nodes=2             # Number of nodes for job
#SBATCH --cpus-per-task=1     # 1 core per task
#SBATCH --ntasks=40           # Each OMPI thread is seen as a single task (so 1 core per thread)
#SBATCH --partition=infinite  # Partition (queue) to submit to

module load gcc/openmpi-1.10.4-hfi

sdir=/scratch/$USER/$SLURM_JOB_ID    # define scratch folder on local disk
mkdir -p $sdir                       # create scratch folder
cp -r $SLURM_SUBMIT_DIR/* $sdir/     # copy all files from cwd to scratch folder
cd $sdir

mpirun ./myjob -i myfile.inp -o myfile.out          # run OMPI job from scratch directory

cp *.out $SLURM_SUBMIT_DIR/                 # copy output back to cwd
cd $SLURM_SUBMIT_DIR
rm -r $tdir                                 # clean up scratch files
```

6 Available Software

To get a list of pre-compiled binaries please use the “module” command. “module avail” shows all software that is installed, or you can also browser the folder “/opt/cluster/programs” if you need to configure your environment further.

7 Compiling Code

Configure your environment using “module” to select the compiler that you need. Note that “hfi” modules are preconfigured to work with Omni-Path communication for fast I/O and inter-node communication across the network.

For GCC/OpenMPI:

```
module load gcc/openmpi-1.10.4-hfi
```