Scientific Computing for Physicists - Teach Cluster

Ramses van Zon

PHY1610H 2026 Winter

What is the Teach cluster?

Teach cluster

(The Teach cluster is just one of these racks)

The Teach cluster is a 2560-core cluster provided for teaching purposes.

It has currently no GPU capability.

It is configured similarly to the coming SciNet production systems Trillium, however it uses hardware repurposed from its predecessor, Niagara.

This system should not be used for production work as the queuing policies are designed to provide fast job turnover and limit the amount of resources one person can use at a time.

https://docs.scinet.utoronto.ca/index.php/Teach

Teach Cluster Specs

- Number of nodes: 64
- CPUs: x86 Intel Skylake
- Cores per node: 40
- Memory per node: 188 GiB
- Scheduler: SLURM
- Infiniband interconnect 1:1
- Parallel shared file system
- Login: teach.scinet.utoronto.ca
- Support: courses@scinet.utoronto.ca

“English, please!”

- Teach consists of 64 “nodes”. You can think of these as individual computers, like your laptop.
- Teach is a “cluster”. This means the nodes are connected to each other with a very fast network (an ”interconnect”). This means that communication between nodes is very fast, allowing for very large computations.
- Each node has
  - 40 “cores” (computation “brains”).
  - About 200 GB of RAM (memory). Your job has to fit in that much memory.
  - No local hard drive.
- The nodes all share the same (non-local) file system.
- The operating system of the nodes is Linux.

Using Teach


SciNet
ADVANCED RESEARCH COMPUTING at the UNIVERSITY OF TORONTO

Access

1. You received an email with your username and a link to the page where you can set your password.
2. Different from our other clusters, you do not need to setup Multi-factor Authentication
3. Setting up SSH keys is optional.

Using Teach: Linux command line

Teach is a Linux-based system:

- This means that the only way to access the system is using the Linux command line.
- There is no Graphical User Interface (GUI).
- As with all SciNet systems, access is via ssh only.
- You must use a terminal program to log in. On a Mac, use the "Terminal" program.
- If you use a Windows machine, download one of the many terminal programs available:
  - MobaXterm (https://mobaxterm.mobatek.net)
  - git bash (https://git-scm.com/downloads)
  - putty (https://putty.org)
  - cygwin (https://cygwin.com)
  - Windows Subsystem for Linux (https://learn.microsoft.com/en-us/windows/wsl/install)

Using the terminal commands requires education which is beyond the scope of this class. Visit the SciNet education site to find classes on the Linux command line.

Access: Logging in

ssh into the Teach login nodes

```
laptop> ssh USERNAME@teach.scinet.utoronto.ca
Password:
teach-login01:~$
```

Login nodes

These are where you develop, edit, compile, prepare and submit jobs.

These nodes are shared, i.e., multiple users are on the same nodes.

These nodes have limits in terms of how long you can run and the memory your applications can use.

Compute nodes

To do computations on Teach, you must submit a batch job. In a job script, you can specify how many compute nodes you need and for how long.

Once the job scheduler starts your job, that is the only thing running on its reserved nodes.

Moving data

Use scp or rsync to and from teach.scinet.utoronto.ca.

E.g.

```
laptop> scp this USERNAME@teach.scinet.utoronto.ca:that
```

```
laptop> rsync this USERNAME@teach.scinet.utoronto.ca:that
```

These commands must be given on your laptop/computer.

Usage: Software and Libraries

Once you are on one of the login nodes, what software is already installed?

- Other than essentials, all installed software is made available using module commands.
- These set environment variables (PATH, etc.)
- Allows multiple, conflicting versions of a given package to be available.
- module overview shows the available software.

```
teach-login01:~$ module overview
----- Cluster specific modules -----
catch2     (1)   methyldackel (1)
palemoon   (1)   rarray (1)
...

-Compiler-dependent avx512 modules -
abseil     (1)   hwloc        (3)
plink      (1)   actc         (1)
....
```

Module subcommands

- `module load <module-name>`

  use particular software
- `module purge`

  remove currently loaded modules
- `module spider`

  (or `module spider <module-name>`)

  list available software packages
- `module list`

  list loaded modules

Usage: Module examples

```
teach-login01:~$ module load openmpi
Lmod has detected the following error:  These module(s) or extension(s) exist but
cannot be loaded as requested: "openmpi"
   Try: "module spider openmpi" to see how to load the module(s).
```

```
teach-login01:~$ module spider openmpi
  openmpi:
-------------------------------------------------------------------------------
    Description:
      The Open MPI Project is an open source MPI-3 implementation
     Versions:
        openmpi/4.0.3
        ...
        openmpi/5.0.8
-------------------------------------------------------------------------------
  For detailed information about a specific "openmpi" module use the full name.
  For example:
     $ module spider openmpi/5.0.8
```

Usage: Module examples, continued

```
teach-login01:~$ module spider openmpi/5.0.8
--------------------------------------------------------------------------------
  openmpi: openmpi/5.0.8
--------------------------------------------------------------------------------
    Description:
      The Open MPI Project is an open source MPI-3 implementation
    You will need to load all module(s) on any one of the lines below before the "openmpi/5.0.8" module i

      StdEnv/2023  gcc/14.3
      StdEnv/2023  intel/2025.2.0
      StdEnv/2023  llvm/21.1.5

    Help:
      Description
      ===========
      The Open MPI Project is an open source MPI-3 implementation.
      More information
      ================
       - Homepage: https://www.open-mpi.org/
```

Usage: Module examples, continued

```
teach-login01:~$ module load gcc/14.3
teach-login01:~$ module load openmpi/5.0.8
```

```
teach-login01:~$ module list
Currently Loaded Modules:
  1) gentoo/2023   (S)   4) hwloc/2.12.1      7) pmix/5.0.8    10) openmpi/5.0.8 (m)
  2) gcccore/.14.3 (H)   5) ucx/1.19.0        8) prrte/3.0.11
  3) gcc/14.3      (t)   6) libfabric/2.1.0    9) ucc/1.4.4
```

Usage: Tips for loading modules

- We advise ***against*** loading modules in your .bashrc file.

  This could lead to very confusing behaviour under certain circumstances.
- Instead, load modules by hand when needed, or by sourcing a separate script.
- Load run-specific modules inside your job submission script.
- Short names give default versions; e.g. `gcc` → `gcc/12.3` (which you do not want).

  It is usually better to be explicit about the versions, for future reproducibility.

Usage: Testing

- Small test jobs can be run on the login nodes.

  Rule of thumb: couple of minutes, taking at most 1-2GB of memory and a couple of cores.
- You can run the the ddt debugger after `module load ddt`.
- The ddt module also gives you the map performance profiler.
- Short tests on Teach that do not fit on a login node, or for which you need a dedicated node, request an interactive debug job with the debugjob command

```
teach-login01:~$ debugjob -n C
```

  where C is the number of cores. The duration of your interactive debug session can be at most four hours.

Usage: Submitting jobs to the Compute Nodes

- Teach uses SLURM as the job scheduler.
- You submit jobs from a login node by passing a script to the sbatch command:

```
teach-login01:~$ sbatch jobscript.sh
```

- This puts the job in the queue. It will run on the compute nodes in due course.
- Jobs will run under their group's RRG allocation, or, if the group has none, under a RAS (or "default") allocation.

Example submission script (Many serial jobs)

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=3:00:00
#SBATCH --job-name serialjob
#SBATCH --output=serial_output_%j.txt
#SBATCH --mail-type=FAIL
module load gcc/14.3

./mycode
```

```
teach-login01$ sbatch serialjob.sh
```

- First line indicates that this is a bash script.
- Lines starting with `#SBATCH` go to SLURM.
- sbatch reads these lines as a job request
- In this case, SLURM looks for one cpu to be run for 3 hours.
- Once it found such a node, script is run:
  - Loads module
  - runs mycode

Usage: Monitoring jobs - command line

Once the job is in the queue, there are some commands you can use to monitor its progress:

- `squeue --me` to show your jobs in the queue (`squeue` for all jobs);
- `squeue -j JOBID` or `scontrol show job JOBID` to get information on a specific job.
- `squeue --start -j JOBID` to get an estimate for when a job will run.
- `jobperf JOBID` to get an instantaneous view of the CPU+memory usage of a running job's nodes.
- `scancel -i JOBID` to cancel the job.
- `scancel -u USERID` to cancel all your jobs (careful!).
- `sinfo -p compute` to look at available nodes.
- `sacct` to get information on your recent jobs.
- SLURM documentation: https://docs.scinet.utoronto.ca/index.php/Slurm