Introduction to SciNet, Niagara & Mist

Erik Spence

SciNet HPC Consortium

8 May 2024

Outline

Today's class will cover

- About SciNet
- Accessing Niagara & Mist
- Using Niagara & Mist
- Data management and file input/output tips

Please stop me if you have any questions.

About SciNet

SciNet is the High-Performance Computing (HPC) consortium at the University of Toronto.

- We run massively parallel computers to meet the needs of researchers across Canada.
- 5 other Canadian consortia provide academic Advanced Research Computing resources.

These consortia maintain a network of resources available to researchers across Canada, under the allocation system run by the Digital Research Alliance of Canada (the "Alliance"):

- Four heterogenous ("general purpose") clusters:
  - ▶ Cedar (Simon Fraser University)
  - ▶ Graham (University of Waterloo)
  - ▶ Béluga & Narval (École de Technologie Supérieure)
- One homogeneous ("large parallel") cluster:
  - ▶ Niagara (University of Toronto)
- One homogeneous GPU cluster:
  - ▶ Mist (University of Toronto)
- Several cloud systems (Sherbrooke, Victoria, Waterloo).

What does SciNet do?

We host one of the fastest supercomputers in Canada available to academics, Niagara.

Plus a bunch of smaller ones:

- Mist (GPU cluster)
- Rouge (GPU cluster)
- Teach (for teaching)

And a long-term storage facility:

- HPSS

What else does SciNet do?

We also teach you how to use our machines.

- Intro to SciNet and Niagara, Linux Shell
- Scientific and Parallel Programming (C, C++, Fortran, R, Python, CUDA)
- Graduate courses on Scientific Computing , Data Analysis, and BioStatistics
- Data management, Parallel I/O, Databases, Machine learning, AI
- Compute Ontario HPC summer school
- International HPC summer school (together with PRACE, XSEDE, RIKEN)
- And others...

For the full list see: `https://scinet.courses`

SciNet people

Software, user support, training, etc.

- Alexey Fedoseev
- Yohai Meiron
- Bruno Mundim
- Mike Nolta
- Erik Spence
- Ramses van Zon
- James Willis
- Chief Technical Officer: Daniel Gruner

Hardware, systems, etc.

- Joseph Chen
- Leslie Groer
- Norbert Krawiec
- Jaime Pinto
- Marco Saldarriaga
- Vladimir Slavnic
- Ching-Hsing Yu

SciNet's staff is here to help you!

Questions? Need help?

- Read the wiki:
  - https://docs.scinet.utoronto.ca/index.php/Niagara_Quickstart
  - https://docs.scinet.utoronto.ca/index.php/Mist
  - https://docs.scinet.utoronto.ca/index.php/Rouge
  - https://docs.alliancecan.ca/wiki/Technical_documentation
- Still need help? Email to **support@scinet.utoronto.ca** or **niagara@tech.alliancecan.ca**.
- Still need help? We'll set up a one-to-one consultation.

Don't be afraid to contact us! We are here to help.

Niagara

This is SciNet main workhorse.

- Total of 80,960 Intel x86-64 cores.
- 2,024 Lenovo SD530 nodes:
  - ▶ 40 SkyLake (or CascadeLake) cores @2.4GHz per node (with hyperthreading to 80 threads).
  - ▶ 188 GiB / 202 GB RAM per node. (at least 4 GiB/core for user jobs)
- 3.6 PFlops sustained / 6.25 PFlops theoretical (#53 on the June 2018 TOP500 list)
- Operating system: Linux (CentOS 7).
- Parallel shared file system.
- Burst buffer for fast I/O.

- Interconnect: InfiniBand Dragonfly+ 1:1 up to 432 nodes, effectively 2:1 beyond that.
- No GPUs, no local disk.

Mist

Mist is Niagara's little GPU sibling.

- 54 IBM Power-9 nodes:
  - ▶ 4 NVIDIA Volta GPUs with 32GB,
  - ▶ 32 Power-9 cores @2.4GHz,
  - ▶ 256 GB RAM per node.
- 1 PFlop peak / 1.6 PFlops theoretical.
- Operating system: Red Hat Enterprise Linux 7.
- Interconnect: 1:1 Infiniband Dragonfly+.
- Same parallel file system as Niagara.

Rouge

A contributed system is 'Rouge'.

- Only available for members of University of Toronto.
- Donated by AMD as part of their COVID-19 HPC Fund.
- 20 x86_64 nodes:
  - ▶ 8 Radeon Instinct MI50 GPUs,
  - ▶ AMD EPYC 7642 48-core CPU,
  - ▶ 512 GB RAM per node.
- Interconnect: 2xHDR100 Infiniband.
- Same parallel file system as Niagara.

To get access to this system, send us an email.

Niagara, for beginners

If you are a beginner the previous three slides might have been difficult to understand. Here it is again, but in simpler language.

- Niagara consists of 2,024 "nodes". You can think of these as individual computers, like your laptop.
- Niagara 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.
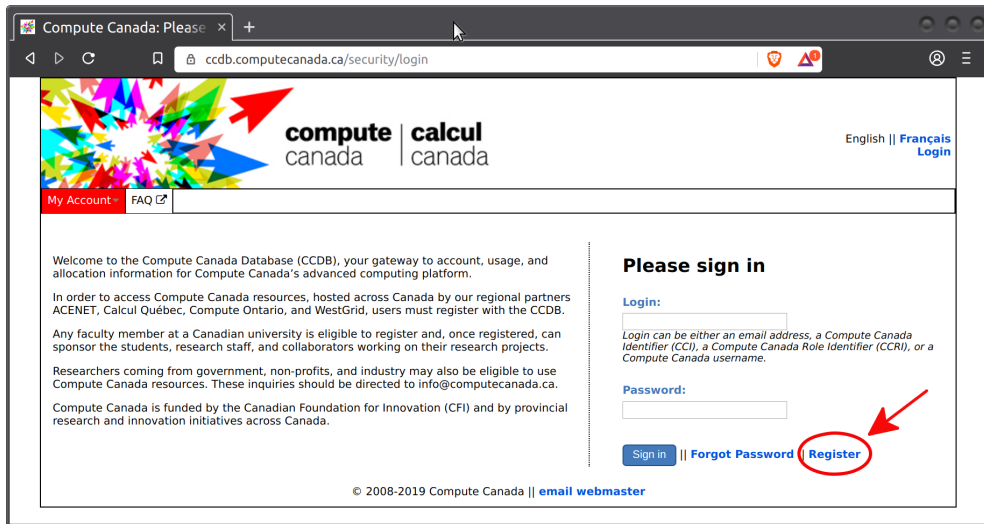- The operating system of the nodes is Linux.

Using Niagara & Mist: Getting access

Getting access to SciNet systems requires registering with the Alliance.

- Registration is done through the Compute Canada DataBase (CCDB) (https://ccdb.alliancecan.ca).
- Access is available to all academic researchers (Principle Investigators) in Canada. Access is free.
- Once the PI has registered with the Alliance a second request must be made for the PI to gain access to SciNet systems.
- Once your PI has access to SciNet systems you may request an Alliance account, and then a SciNet account. You will need your PI's Compute Canada Role Identifier (CCRI) to register.
- The process can take a few days, under normal circumstances.

Registering with the Alliance allows us to track your group's usage across the country.

Using Niagara & Mist: CCDB registration

Using Niagara: Linux command line

Niagara 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).
- 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)

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. There will be a class on the Linux command line given a week today.

Using Niagara: SSH keys

As with all SciNet and Alliance systems, access to Niagara is via SSH (secure shell) only. This allows a secure connection to be established between your local machine and the Niagara login nodes.

Not only must SSH be used to connect to Niagara, but SSH keys must also be used. Passwords are not accepted!

Using SSH keys involves creating a public-private key pair. How you do this depends on the particular terminal or client that you're using to access the systems. See the instructions on the SciNet or Alliance documentation sites.

Once you have created your keys, upload your public key to the CCDB.

Using Niagara: MFA

Since April 2024, Multi-Factor Authentication (MFA) is mandatory for accessing all Alliance systems, including SciNet systems.

There are two MFA options:

- Duo (an app which you register with your phone).
- Yubikey (a hardware device used for second factor authentication).

Instructions for setting up MFA can be found here:

`https://docs.alliancecan.ca/wiki/Multifactor_authentication`

Using Niagara: Logging in

Once you have your Alliance and SciNet account, and you have

- created your SSH keys,
- uploaded your public key to CCDB,
- set up your multifactor authentication,

you are ready to log into Niagara.

```
ejspence@mycomp ~> ssh -Y ejspence@niagara.scinet.utoronto.ca
(ejspence@niagara.scinet.utoronto.ca) Duo two-factor login for ejspence

Enter a passcode or select one of the following options:

1.  Duo Push to My phone (Android)

Passcode or option (1-1):  1
Success.  Logging you in...
ejspence@nia-login07 ~>
```

Using Niagara: Login nodes

Once you have logged in, you are on a login node.

- The Niagara login nodes are where you develop, edit, compile, prepare and submit jobs.
- These login nodes are not part of the Niagara compute cluster, but have the same architecture, operating system, and software stack.
- The optional -Y is needed to open windows from the Niagara command-line onto your local X server.
- Short testing of your jobs on login nodes is permissible, as long as they only use a few cores, and don't last very long.
- To log into Mist, replace Niagara with Mist, above.
- To log into Rouge, login to Niagara first, then SSH to the Rouge login node.

Login nodes are for setting up your jobs. Do not run production work on the login nodes!

Storage Systems and Locations

When you log in you will automatically be in your home directory.

You have a home and scratch directory on Niagara, whose locations are given by

$HOME=/home/g/groupname/myusername

$SCRATCH=/scratch/g/groupname/myusername

```
ejspence@nia-login07 ~>
ejspence@nia-login07 ~> pwd
/home/s/scinet/ejspence
ejspence@nia-login07 ~> cd $SCRATCH
ejspence@nia-login07 ejspence>  pwd
/scratch/s/scinet/ejspence
ejspence@nia-login07 ejspence>
```

Users from groups with a RAC allocation may also have a project directory.

$PROJECT=/project/g/groupname/myusername

IMPORTANT: Future-proof your scripts Use environment variables (HOME, SCRATCH, PROJECT) instead of the actual paths! **The paths may change in the future.**

Storage Limits on Niagara

location	quota	# files	expiration	backed up	on compute
$HOME	100 GB	250K		yes	read-only
$SCRATCH	25 TB	6M	2 months	no	yes
$PROJECT	by group allocation			yes	yes
$ARCHIVE	by group allocation			dual-copy	no
$BBUFFER	10TB, by request		48 hours	no	yes

- Compute nodes do not have local storage.
- Running jobs should save to $SCRATCH.
- Archive space (long term storage) is on HPSS.
- Backup means a recent snapshot, not an achive of all data that ever was.
- $BBUFFER stands for the Burst Buffer. This is a special file system for high file Input/Output jobs. Access is by request.

Moving data

Move amounts less than 10GB through the login nodes.

- Compute nodes are not visible from outside the SciNet data centre.
- Use scp or rsync to niagara.scinet.utoronto.ca or niagara.alliancecan.ca (same thing).
- This will time out for amounts larger than about 10GB.

Move amounts larger than 10GB through the datamover nodes.

- You can also directly transfer to these nodes from the outside world
  (nia-datamover1.scinet.utoronto.ca, or nia-datamover2).
- If you do this often, consider using Globus, a web-based tool for data transfer.

Moving data to HPSS/Archive/Nearline using the scheduler.

- HPSS is a tape-based storage solution.
- Storage space on HPSS is allocated through the annual Alliance RAC allocation.

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 spider" searches for available software.

```
ejspence@nia-login07 ~> module spider
---------------------------------------------------
The following is a list of the modules currently av
---------------------------------------------------
CCEnv:  CCEnv
NiaEnv:  NiaEnv/2018a
autotools:  autotools/2017
autoconf, automake, and libtool
boost:  boost/1.66.0
cfitsio:  cfitsio/3.430
cmake:  cmake/3.10.2 cmake/3.10.3


...
```

Software and Libraries, continued

There are a whole variety of *module* commands for controlling what software is currently available. Some of the important commands are:

- *module load <module-name>*
  load a particular software version

- *module purge*
  remove all currently loaded modules

- *module spider* (or *module spider <module-name>*)
  search for software packages

- *module avail*
  list loadable software packages

- *module list*
  list currently loaded modules

Software and Libraries, continued more

On Niagara, there are really two software stacks:

- A Niagara software stack tuned and compiled specifically for this machine. This stack is available by default, but if not, can be reloaded with

```
ejspence@nia-login07 ~> module load NiaEnv
```

- The same software stack available on the Alliance General Purpose clusters Graham, Cedar, and Béluga.

```
ejspence@nia-login07 ~> module load CCEnv
```

If you want the same default modules loaded as on Cedar and Graham, then afterwards also *module load StdEnv*.

Note that the CC stack will not work on Mist, as Mist is a Power-9 system.

Tips for loading software

We suggest you follow these software-loading conventions.

- We advise **against** loading modules in your .bashrc. 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. intel → intel/2019u4. It is better to be explicit about the versions, for future reproducibility.
- Modules often require other modules to be loaded first. Solve these dependencies by using *module spider*.

Module spider

Oddly named, the module subcommand spider is the search-and-advice facility for modules.

```
ejspence@nia-login07 ~> module load openmpi
Lmod has detected the error:  These module(s) exist but cannot be loaded as requested:  "openmpi"
Try:  "module spider openmpi" to see how to load the module(s).
ejspence@nia-login07 ~>
ejspence@nia-login07 ~> module spider openmpi
--------------------------------------------------------------------------------------------
openmpi:
--------------------------------------------------------------------------------------------
Versions:
openmpi/3.1.3
openmpi/4.0.1
openmpi/4.0.3
--------------------------------------------------------------------------------------------
For detailed information about a specific "openmpi" module (including how to load the modules)
use the module's full name.  For example:
$module spider openmpi/4.0.3
```

Module spider, continued

```
ejspence@nia-login07 ~> module spider openmpi/4.0.3
-------------------------------------------------------------------------------------------
openmpi:  openmpi/4.0.3
-------------------------------------------------------------------------------------------
You will need to load all module(s) on any one of the lines below before the "openmpi/3.1.0rc3"
module is available to load.

gcc/9.2.0
ejspence@nia-login07 ~>
ejspence@nia-login07 ~> module load gcc/9.2.0
ejspence@nia-login07 ~> module load openmpi/4.0.3
ejspence@nia-login07 ~>
ejspence@nia-login07 ~> module list
Currently Loaded Modules:  1) NiaEnv/2019b 2) gcc/9.2.0 3) openmpi/4.0.3
ejspence@nia-login07 ~>
```

Text editors on Niagara

You've brought your data and your code over to Niagara. Now you need to make some modifications to your code. For that you need a text editor.

Niagara has all of the usual Linux text editors available:

- Emacs
- vi/vim
- nano
- VSCode

Note that these are best run within the terminal (other than VSCode), rather than opening a window (such as XEmacs). This uses much less network traffic, and will be significantly faster.

Can I Run Commercial Software?

Some researchers use commercial software. Can you run such software on Niagara?

- Possibly, but you have to bring your own license for it.
- SciNet and Compute Canada have an extremely large and broad user base of thousands of users, so we cannot provide licenses for everyone's favorite software.
- Thus, the only commercial software installed on Niagara is software that can benefit everyone: Intel compilers, math libraries and parallel debuggers.
- That means no MATLAB, Gaussian, IDL, . . .
- Open source alternatives like Octave, Python, R are available.
- We are happy to help you to install commercial software for which you have a license.
- In some cases, if you have a license, you can use software in the Compute Canada software stack.

Python and R modules

Some researchers use Python and R. How is this managed?

- Python and R are available as modules.
- Other than the very basics, packages for Python and R are not installed as modules under the Niagara stack. These should be installed in your own home directores.
  - ▶ For R, just use *install.packages(...)*.
  - ▶ For Python, create a virtual environment and install your packages there.
  - ▶ If you are using Mist, load the anaconda3 module and create a conda environment (see the instructions on the SciNet wiki).

```
ejspence@nia-login07 ~> module load python/3.11.5
ejspence@nia-login07 ~> virtualenv --system-site-packages myenv
ejspence@nia-login07 ~> source myenv/bin/activate
(myenv)ejspence@nia-login07 ~> pip install myawesomepackage
(myenv)ejspence@nia-login07 ~>
```

Compiling on Niagara: Example

```
ejspence@nia-login07 ~>
ejspence@nia-login07 ~> module list
Currently Loaded Modules:
1) NiaEnv/2019b
ejspence@nia-login07 ~>
ejspence@nia-login07 ~> module load intel/2019u4 gsl/2.5
ejspence@nia-login07 ~>
ejspence@nia-login07 ~> ls
main.c module.c
ejspence@nia-login07 ~> icc -c -O3 -xHost -o main.o main.c
ejspence@nia-login07 ~> icc -c -O3 -xHost -o module.o module.c
ejspence@nia-login07 ~> icc -o main module.o main.o -lgsl -mkl
ejspence@nia-login07 ~>
ejspence@nia-login07 ~> ./main
```

Testing

You really should test your code before you submit it to the cluster to know if your code is correct and what kind of resources you need.

- Small test jobs can be run on the login nodes. Rule of thumb: couple of minutes, taking at most about 1-2GB of memory, couple of cores.
- You can run the the DDT (Forge) debugger on the login nodes after *module load ddt*.
- Short tests 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

```
ejspence@nia-login07 ~> debugjob N
```

where N is the number of nodes. The duration of your interactive debug session can be at most one hour, can use at most N=4 nodes, and each user can only have one such session at a time.

Submitting jobs to Niagara

It's time to submit jobs!

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

```
ejspence@nia-login07 ~> 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 allocation (previously called 'default' allocation).
- Scheduling is by node, on Niagara, so in multiples of 40-cores (use all the cores!).
- Scheduling is by GPU, on Mist (use all the GPUs!).
- Maximum walltime is 24 hours.
- Data must be writen to SCRATCH or PROJECT (HOME is read-only on compute nodes).
- Compute nodes have no internet access. Download data you need beforehand.

Scheduling by Node

- All job resource requests on Niagara are scheduled as a multiple of **nodes**.
- The nodes that your jobs run on are exclusively yours.
  - ▸ No other users are running anything on them.
  - ▸ You can SSH into them to see how things are going.
- Whatever your requests to the scheduler, it will always be translated into a multiple of nodes.
- Memory requests to the scheduler are of no use. Your job gets N x 202GB of RAM if N is the number of nodes.
- You must **use all 40 cores on each of the nodes** that your job uses. You will be contacted if you don't, and we will help you get more science done.
- The exception is if you cannot use all cores because your job uses too much memory.
- You are granted 8 cores for each GPU on Mist. You are expected to use the GPU as much as possible. The cores are less important.

Example submission script (OpenMP)

```bash
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --cpus-per-task=40
#SBATCH --time=1:00:00
#SBATCH --job-name openmp_job
#SBATCH --output=openmp_output_%j.txt


cd $SLURM_SUBMIT_DIR


module load intel/2019u4


export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK


./openmp_example # or 'srun ./openmp_example'
```

```
ejspence@nia-login07 ~> sbatch openmp_job.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 (which it gives the name openmp_job).
- In this case, SLURM looks for one node with 40 cores to be run with one task, for 1 hour.
- Once it finds such a node, it runs the script:
  - ▶ Change to the submission directory;
  - ▶ Loads modules;
  - ▶ Sets an environment variable;
  - ▶ Runs the **openmp_example** application.

Example submission script (Python)

```bash
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --cpus-per-task=40
#SBATCH --time=1:00:00
#SBATCH --job-name serial

cd $SLURM_SUBMIT_DIR

module load python/3.6.8

(cd job1 && python ../myscript 1 && echo "job 1 done") &
(cd job2 && python ../myscript 2 && echo "job 2 done") &
⋮
(cd job39 && python ../myscript 39 && echo "job 39 done") &
(cd job40 && python ../myscript 40 && echo "job 40 done") &
wait
```

- This is a bash script.
- In this case, SLURM looks for one node with 40 cores for 1 hour.
- Once it finds such a node, it runs the script:
  - ▶ Change to the submission directory;
  - ▶ Loads modules;
  - ▶ Runs the 40 Python processes and waits until they're done.

Monitoring jobs

Once the job is incorporated into the queue, there are some commands you can use to monitor its progress.

- squeue to show the job queue ("squeue -u $USER" for just your jobs);
- "squeue -j JOBID" to get information on a specific job (alternatively, scontrol show job JOBID, which is more verbose).
- "squeue --start -j JOBID" to get an estimate for when a job will run.
- "jobperf JOBID" gives the instantaneous view of the cpu+memory usage of a running job.
- "scancel JOBID" to cancel the job, or "scancel -u USERID" to cancel all your jobs (careful!).
- "sacct" to get information on your recent jobs.

The website my.scinet.utoronto.ca is very useful for monitoring present and past job performance.

Monitoring jobs, continued

Use `my.scinet.utoronto.ca`! All of the information about your past job performance is available there.

- Niagara CPU and storage utilization.
- Status of the login nodes.
- Niagara and Mist job history.
- Per job:
  - jobscript,
  - wall time,
  - memory, CPU, disk I/O every 10 minutes
  - and much more!

This is a very helpful resource for monitoring your job performance, and diagnosing problems.

Data Management and I/O Tips

Use the shared file system carefully!

- $HOME, $SCRATCH, and $PROJECT all use the parallel file system called GPFS.
- Your files can be seen on all Niagara and Mist login and compute nodes.
- GPFS is a high-performance file system which provides rapid reads and writes to large data sets in parallel from many nodes.
- But accessing data sets which consist of many, small files leads to poor performance.
- Avoid reading and writing lots of small amounts of data to disk.
- Many small files on the system would waste space and would be slower to access, read and write.
- Write data out in binary. Faster and takes less space.
- Burst buffer is better for I/O heavy jobs and to speed up checkpoints.

Further information

Useful sites:

- SciNet: `https://www.scinet.utoronto.ca`
- Niagara: `https://docs.scinet.utoronto.ca/index.php/Niagara_Quickstart`
- Mist: `https://docs.scinet.utoronto.ca/index.php/Mist`
- Rouge: `https://docs.scinet.utoronto.ca/index.php/Rouge`
- System Status: `https://docs.scinet.utoronto.ca`
- Training: `https://scinet.courses`
- My.SciNet: `https://my.scinet.utoronto.ca`
- Alliance documentation:
  `https://docs.alliancecan.ca/wiki/Technical_documentation`

Furtherer information

Some specific help:

- Creating an SSH key-pair:
  https://docs.scinet.utoronto.ca/index.php/SSH#SSH_Keys
- Multi-factor authentication:
  https://docs.scinet.utoronto.ca/index.php/Two-Factor_setup
- Linux command line class: https://scinet.courses/1320

Support:

- support@scinet.utoronto.ca
- niagara@tech.alliancecan.ca