Introduction to SciNet, Niagara & Mist

The SciNet Team

April 10, 2024

Outline

What we will cover:

- About SciNet
- Accessng Niagara and Mist
- Using Niagara and Mist

Feel free to ask questions during the presentation.

About SciNet

About SciNet

SciNet is a centre for high-performance computing at the University of Toronto.

- We run massively parallel computers to meet the needs of researchers across Canada.
- 5 other HPC centres in Canada also provide academic **Advanced Research Computing** resources.
- These centres maintain and support a network of resources available to researchers across Canada, under a national allocation system administered by the **Digital Research Alliance of Canada.**

National research computing clusters

- Four general purpose clusters:
  - Cedar (Simon Fraser University)
  - Graham (University of Waterloo)
  - Béluga (École de technologie supérieure)
  - Narval (École de technologie supérieure)
- One large parallel CPU cluster:
  - Niagara (University of Toronto)
- One homogeneous GPU cluster:
  - Mist (University of Toronto)
- Several cloud systems (Sherbrooke, Victoria, Waterloo).

What does SciNet do?

https://www.scinet.utoronto.ca

Systems

We host one of the largest supercomputers in Canada available to academics.

- Niagara CPU cluster

Plus some smaller ones

- Mist GPU cluster
- Teach
- Rouge
- S4H
- Balam GPU cluster

And a longer-term storage facility

- HPSS (a.k.a. archive a.k.a. nearline)

What else does SciNet do?

Training

- Intro to SciNet and Niagara, Linux Shell
- Scientific and Parallel Programming (Python, C++, R, GPU programming)
- Grad 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 HPC organizations abroad)

For full list see: https://tinyurl.com/scinetallcourses

SciNet people

Software, user support, training, etc..

- Mike Nolta
- Erik Spence
- Ramses van Zon
- Bruno Mundim
- Alexey Fedoseev
- James Willis
- Yohai Meiron

- Chief Technical Officer: Daniel Gruner
- Associate CTO: Joseph Chen
- Operations Coordinator: Leanne De Guia

Hardware, systems, etc..

- Ching-Hsing Yu
- Leslie Groer
- Jaime Pinto
- Marco Saldarriaga
- Vladimir Slavnic
- Ram Sharma
- Norbert Krawiec

- Information Systems Security: Shawn Winnington-Ball

SciNet's staff is here to help you.

Questions? Need help?

Check the wiki pages:

- Niagara: https://docs.alliancecan.ca/wiki/Niagara_Quickstart
- Mist: https://docs.scinet.utoronto.ca/index.php/Mist
- Other Alliance clusters or general topics: https://docs.alliancecan.ca

Still need help?

- Email to **niagara@tech.alliancecan.ca** or **support@scinet.utoronto.ca**
- Still need help? Request a one-to-one consultation (request via email).

Niagara

- 80,960 x86-64 cores.
- 2,024 *Lenovo SD530* nodes
- Per node:
  - 40 Intel SkyLake/CascadeLake cores @ 2.4GHz
  - 188 GiB RAM
- 3.6 PFlops sustained (6.25 PFlops theoretical).
  #59 on the Nov 2018 TOP500 (#212 as of Nov 2023)
- InfiniBand Dragonfly+ network
  1:1 up to 432 nodes, 2:1 beyond that.
- Parallel shared file system for home, scratch, project
- Login: **niagara.alliancecan.ca**
  Support: **niagara@tech.alliancecan.ca**

Mist

- Niagara's little GPU sibling
- 54 IBM Power-9 nodes with 4 GPUs.
- Per node:
  - 32 Power-9 cores @ 2.4GHz
  - 256 GB RAM per node
  - 4 NVIDIA "Volta" GPUs with 32GB
- 1 PFlops peak (1.6 PFlops theoretical).
- Interconnect: 1:1 InfiniBand Dragonfly+
- Same parallel shared file systems as Niagara
- Login: **mist.alliancecan.ca**
  Support: **niagara@tech.alliancecan.ca**

Niagara, for beginners

If you are a beginner the previous two 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 and Mist

Access

1. Register with the Alliance CCDB

   https://ccdb.alliancecan.ca/account_application

   PIs have to get an account one first, so they can sponsor your account at no cost.

   The approval process typically takes 1-2 business days.
2. Setup Multi-factor Authentication

   https://docs.alliancecan.ca/wiki/Multifactor_authentication
3. Go to

   https://ccdb.alliancecan.ca/services/opt_in

   and click on the "Join" button next to Niagara and Mist.
4. After a business day or two, you get an email confirming your access to Niagara and Mist.
5. Setup SSH keys

Using Niagara: Linux command line

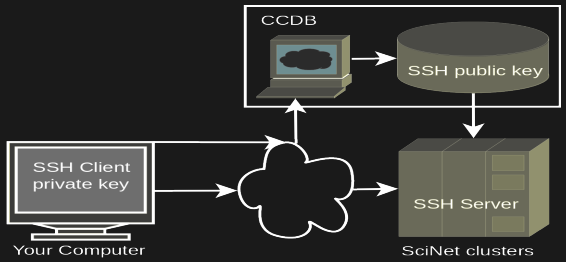
Niagara is a Linux-based system:

- 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)
  - 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: Secure Login

- As with all SciNet and Alliance systems, access is via ssh only.
- The connection will get you to a Linux command line interface.
- Password doesn't work on Niagara and Mist!
  SSH keys must be used to authenticate.
- SSH keys come in a pair:
  - a **private key** which is kept on your own computer and used to connect
  - a **public key** that you upload to CCDB and which then propagates to the clusters.
- After SSH-key authentication, a second factor authentication must happen. (2FA/MFA)



Before Access: Setup Multifactor Authentication

Multifactor authentication (MFA) protects your account even if your password or ssh key is compromised.

Once setup, you will need to perform a second *action* to access some of our services.

You can choose any of these factors for this second authentication step:

- Approve a notification on a smart device through the Duo Mobile application.
- Enter a code generated on demand in the Duo application.
- Push a button on a hardware key (e.g. YubiKey).

MFA is mandatory for SSH authentication starting **April 15, 2024**.

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

Before Access: SSH key setup

- To access SciNet systems for the first time, open a local terminal window on your computer (e.g. MobaXTerm on Windows).
- Then generate an **ssh key pair** with the following command:

```
laptop> ssh-keygen -t ed25519 -C "USERNAME@MYLAPTOP dra" -f ~/.ssh/dra_ed25519
```

- Enter a **passphrase** to protect your private key ("third factor").
- A private key, `dra_ed25519`, and a public key, `dra_ed25519.pub` are then created in the directory "`.ssh`" in your home directory.
- `-f` option specifies the filename of the key file.
- (optional) `-C` option allows you to insert a comment into the key.

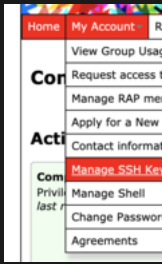
Before Access: Uploading Your Public Key

Once you created your ssh key pair, you need to make Niagara/Mist aware of the public part of your key.

- Step 1: Use your Alliance/CCDB credentials to visit the following site:

  https://ccdb.alliancecan.ca/ssh_authorized_keys

or via the CCDB menu:

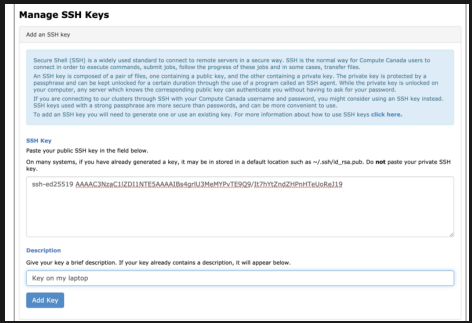


Before Access: Uploading Your Public Key

- Step 2: Grab your SSH public key:

```
laptop> cat ~/.ssh/dra_ed25519.pub
ssh-ed25519 AAAAC3NzaC11ZDI1NTE5AAAAIEpDf+Wcvtru6pUcBgJQo/3+cmI4+MisfNE3U46/CDkx
USERNAME@MYLAPTOP dra
```

- Step 3: Paste the public key into the CCDB form and click “Add Key” button:



Access: Logging in

Wait a few minutes for your newly uploaded public key to propagate to the systems.

Then ssh into the Niagara login nodes specifying the corresponding ssh private key:

```
laptop> ssh -i ~/.ssh/dra_ed25519 USERNAME@niagara.alliancecan.ca
Enter passphrase for dra_ed25519:
nia-login07:~$
```

Instead of **niagara.alliancecan.ca**, you can also use **niagara.scinet.utoronto.ca**.

Perform second factor action (i.e. phone or hardware key push, enter one time code)

Note:

- `-i` option selects a file from which the identity (private key) for key authentication is read.
- For *Mist*, replace *niagara* with *mist*.

SSH key best practices

- Do not share or copy your private key.
- Always protect it with a strong passphrase.
- Create a separate key pair for each computer you use to access our systems.
- Do not create key pairs on shared systems like HPC clusters.

More information:

A reference to help you troubleshooting:
https://docs.alliancecan.ca/wiki/SSH_Keys

If you get a warning "REMOTE HOST IDENTIFICATION HAS CHANGED", check the fingerprints:
https://docs.alliancecan.ca/wiki/SSH_host_keys.

Connecting more conveniently: SSH options and keys

Once you've logged in successfully, you can save the ssh options in `~/.ssh/config`:

```
Host niagara
   HostName niagara.alliancecan.ca
   User USERNAME
   IdentityFile ~/.ssh/dra_ed25519
   IdentitiesOnly yes
```

Now you can access Niagara by simply typing (in addition to your passphrase):

```
laptop> ssh niagara
```

This will also make data transfer commands like `scp` and `rsync` work more easily.

You can use the *ssh-agent* to hold your key for you by typing:

```
laptop> ssh-add ~/.ssh/dra_ed25519
```

This will ask for the passphrase, and then save that key so you do not have to type the passphrase again during the session.

Usage: login, compute, and datamovers nodes

There are three types of nodes on Niagara:

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 Niagara, 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.

Datamover nodes

Meant for large data transfers from, to or on Niagara.

Usage: Storage Systems and Locations

Home and scratch

`$HOME=/home/g/groupname/username`

`$SCRATCH=/scratch/g/groupname/username`

Use these convenient variables!

```
nia-login07:~$ pwd
/home/s/scinet/myusername

nia-login07:~$ cd $SCRATCH

nia-login07:myusername$ pwd
/scratch/s/scinet/myusername
```

Project

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

`$PROJECT=/project/g/groupname/username`

Burst Buffer

Groups with heavy I/O can request access to a smaller, faster parallel file system called burst buffer.
`$BBUFFER=/bb/g/groupname/username`

Usage: Storage Limits

location	quota	#files	block size	expiration	backed up	on login	compute
$HOME	100 GB	250K	1 MB		yes	yes	read-only
$SCRATCH	25 TB	6M	16 MB	2 months	no	yes	yes
$PROJECT	by group allocation	depends	16 MB		yes	yes	yes
$BBUFFER	10TB, by request		1 MB	48 hours	no	yes	yes
$ARCHIVE	by group allocation				dual-copy	no	no

- Compute nodes do not have local storage, but they have a lot of memory, which you can use as if it is local disk (`$SLURM_TMPDIR`)
- $ARCHIVE space, also called nearline storage or HPSS, is not mounted on login or compute nodes.
- Storage space on project and HPSS is allocated through the annual resource allocation competition (RAC).
- Backup means a recent snapshot, not an archive of all data that ever was.

Moving data

To move amounts less than 20GB, use the login nodes

Use scp or rsync to and from niagara.alliancecan.ca.

- For scp to use your ssh key, give it the '`-i ~/.ssh/YOURKEY`' option. *E.g.*

```
laptop> scp -i ~/.ssh/dra_ed25519 this USERNAME@niagara.alliancecan.ca:that
```

- For rsync to use your ssh key, use the '-e "ssh -i ~/.ssh/YOURKEY" ' option. *E.g.*

```
laptop> rsync -e "ssh -i ~/.ssh/YOURKEY" this USERNAME@niagara.alliancecan.ca:that
```

These commands must be given on your computer.

To move amounts larger than 20GB, use the datamover nodes.

- Use scp or rsync with nia-datamover1.scinet.utoronto.ca or nia-datamover2.scinet.utoronto.ca .
- If you do this often, consider using Globus, a web-based tool for data transfer.

To move data to HPSS/Archive/Nearline

- HPSS is a tape-based storage solution, and is SciNet's nearline a.k.a. archive facility.
  - Store and recall using scheduled jobs or Globus.

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.

```
nia-login07:~$ module overview
----- /scinet/niagara/.../base -----
apptainer     (2)   meson         (1)
arm-forge     (1)   mii           (1)
autotools     (2)   mpfr          (1)
bedtools      (2)   muscle        (1)
bowtie2       (1)   namd          (1)
bwa-mem2      (1)   nano          (1)
bwa           (1)   ncl           (1)
cereal        (1)   ncview        (1)
cget          (1)   ninja         (1)
...
----- /scinet/niagara/stacks ------
CCEnv (1)   NiaEnv (3)
```

Usage: Software and Libraries, continued

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

On Niagara, there are a few distinct software stacks whose modules do not mix.

```
module load NiaEnv/2022a
module load NiaEnv/2019b

module load CCEnv StdEnv
```

Stacks compiled for Niagara (newer)
(default, but old)

Software stack as on the Alliance's general purpose clusters.

On Mist, there is one, system-specific stack, with modules like cuda, pgi, xl.

Usage: Module examples

```
nia-login07:~$ 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).
```

```
nia-login07:~$ module spider openmpi
  openmpi:
------------------------------------------------------------------------------
    Description:
      The Open MPI Project is an open source MPI-2 implementation
     Versions:
        openmpi/3.1.3
        openmpi/4.0.1
        openmpi/4.0.3
------------------------------------------------------------------------------
  For detailed information about a specific "openmpi" module use the full name.
  For example:
     $ module spider openmpi/4.0.3
```

Usage: Module examples, continued

```
nia-login07:~$ module spider openmpi/4.0.1
----------------------------------------------------------------------------
  openmpi: openmpi/4.0.1
----------------------------------------------------------------------------
    Description:
      The Open MPI Project is an open source MPI-2 implementation
    You will need to load all module(s) on any one of the lines below before the "ope
      gcc/8.3.0
      gcc/9.2.0
      intel/2019u3
      intel/2019u4
    Help:
      Description
      ===========
      The Open MPI Project is an open source MPI-2 implementation.
      More information
      ================
       - Homepage: https://www.open-mpi.org/
```

Usage: Module examples, continued

```
nia-login07:~$ module load intel/2019u4
nia-login07:~$ module load openmpi/4.0.1
```

```
nia-login07:~$ module list
Currently Loaded Modules:
  1) NiaEnv/2019b (S)   2) intel/2019u4   3) openmpi/4.0.1
```

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. `intel` → `intel/2019u4`.

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

Can I Use Commercial Software?

- Possibly, but you have to bring your own license for it.
- SciNet and the Digital Research Alliance of Canada have an extremely large and broad user base of thousands of users, so we cannot provide licenses for everyone's favourite software.
- Thus, the only commercial software installed on Niagara is software that can benefit everyone, i.e. compilers and debuggers.
- Open source alternatives like Octave, Python, R, Julia are available.
- We are happy to help you to install commercial software for which you have a license.

Usage: Python modules

- Several python versions are available as modules.
- These comes with optimized Numpy, SciPy, ...
- Further packages for Python and R are not installed in modules;
  These need to be installed in users' home directories.
- For installing packages for Python, use virtual environments:

```
nia-login07:~$ module load python/3.9.8
nia-login07:~$ virtualenv --system-site-packages ~/myenv
nia-login07:~$ source ~/myenv/bin/activate
(myenv) nia-login07:~$ pip install THISPACKAGE
```

If you want, use the "venv2jup" command to use your virtual environment in the JupyterHub.

If at all possible, do not use conda environments.

Usage: R modules

- Several R versions are available as modules, but you first need to load a gcc module

```
$ module load gcc
$ module -r avail ^r/
------------- /scinet/niagara/software/2019b/modules/gcc-8.3.0 -------------
 r/3.5.3  r/3.6.1  r/3.6.3 (D)  r/4.0.3  r/4.1.2  r/4.2.2-batteries-included

$ module load r/4.2.2
```

- To install R packages, use the R command "install.packages(...)"
- The first time you do this, you'll be asked if you are okay with installing in your home directory (hint: you are).

Usage: Compiling on Niagara

Suppose you have to compile your own C, C++ or Fortran code.

- Not a problem: Niagara has GNU compilers as well as Intel compilers installed in **modules**.
- Need an MPI library? Not a problem either: Niagara has openmpi and intelmpi libraries as **modules**.
- We recommend that you use the intel compilers with openmpi libraries.
- Use `-march=native` (gcc) or `-xhost` (intel) compilation flags to get the most out of Niagara's CPUs.
- Need libraries? "module load" them.

Example

```
nia-login07:~$ module load intel/2019u4 gsl/2.5
nia-login07:~$ ls
main.c module.c
nia-login07:~$ icc -c -O3 -xHost -o main.o main.c
nia-login07:~$ icc -c -O3 -xHost -o module.o module.c
nia-login07:~$ icc -o main module.o main.o -lgsl -mkl
```

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, couple of cores, ≤ 1 GPU.

- You can run the the ddt debugger after `module load ddt`.
- The ddt module also gives you the map performance profiler.
- Short tests on Niagara 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

```
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.

- For short single-GPU tests on Mist use

```
mist-login01:~$ debugjob -g 1
```

Usage: Submitting jobs to the Compute Nodes

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

```
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 (or "default") allocation.

Keep these in mind when submitting jobs

- Niagara scheduling is by node, so in multiples of 40-cores. *Use all cores!*
- Mist scheduling is by single GPU or by whole node (multiple of 4 GPUs). *Use all GPUs!*
- Maximum walltime is 24 hours.
- Jobs must write to your scratch or project directory (home is read-only on compute nodes).
- Compute nodes have no internet access.

Example submission script (Many serial jobs)

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=40
#SBATCH --time=3:00:00
#SBATCH --job-name serialjob
#SBATCH --output=serial_output_%j.txt
#SBATCH --mail-type=FAIL
module load NiaEnv/2019b
module load python/3
module load gnu-parallel

source ~/myenv/bin/activate
parallel python serial.py ::: {0..99}
```

```
nia-login07:scratch$ 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 node with 40 tasks to be run for 3 hours.
- Submit from /scratch, as /home is read-only.
- Once it found such a node, script is run:
  - Loads modules
  - Activates python environment
  - Has gnu-parallel load-balance 99 tasks over 40 cores.

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

Example submission script (OpenMP)

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --cpus-per-task=40
#SBATCH --time=1:00:00
#SBATCH --job-name omp_job
#SBATCH --output=omp_output_%j.txt
#SBATCH --mail-type=FAIL
module load NiaEnv/2019b intel/2019u4

OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
export OMP_NUM_THREADS

./omp_example # or 'srun ./omp_example'
```

```
nia-login07:scratch$ sbatch omp_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 `omp_job`).
- In this case, SLURM looks for one node with 40 cores to be run inside one task, for 1 hour.
- Submit from /scratch, as /home is read-only.
- Once it found such a node, script is run:
  - Loads modules;
  - Sets an environment variable;
  - Runs the `omp_example` application.

Example submission script (MPI)

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=40
#SBATCH --time=3:00:00
#SBATCH --job-name mpi_job
#SBATCH --output=mpi_output_%j.txt
#SBATCH --mail-type=FAIL
module load NiaEnv/2019b
module load intel/2019u4
module load openmpi/4.0.1

mpirun ./mpi_app # or 'srun ./mpi_app'
```

```
nia-login07:scratch$ sbatch mpi_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 `mpi_job`)
- In this case, SLURM looks for 2 nodes with 40 cores on which to run 80 tasks, for 3 hours.
- Submit from /scratch, so output can be written.
- Once it found nodes, the script is run:
  - Loads modules;
  - Runs the `mpi_app` application.

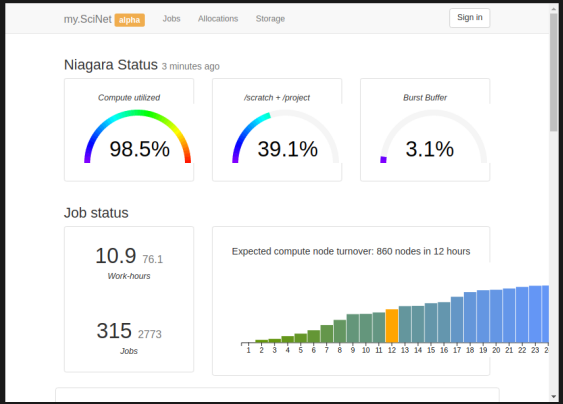
Usage: Monitoring jobs - command line

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

Usage: my.scinet.utoronto.ca

Check out https://my.scinet.utoronto.ca for past and present job info.

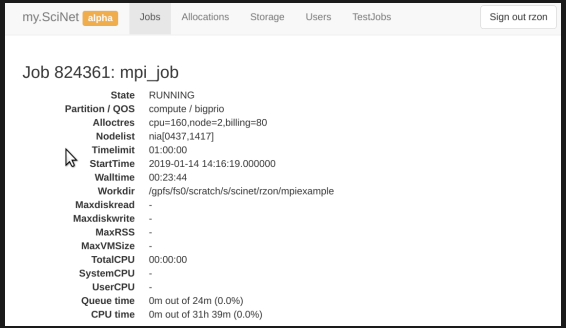


Features

- Niagara CPU and storage utilization
- Status of the login nodes
- Niagara and Mist job history
- Per job:
  - jobscript
  - environment
  - wall time
  - memory usage every 10 minutes.
  - CPU usage every 10 minutes.
  - GFlops/s every 10 minutes.
  - disk I/O usage every 10 minutes.

Usage: my.scinet.utoronto.ca

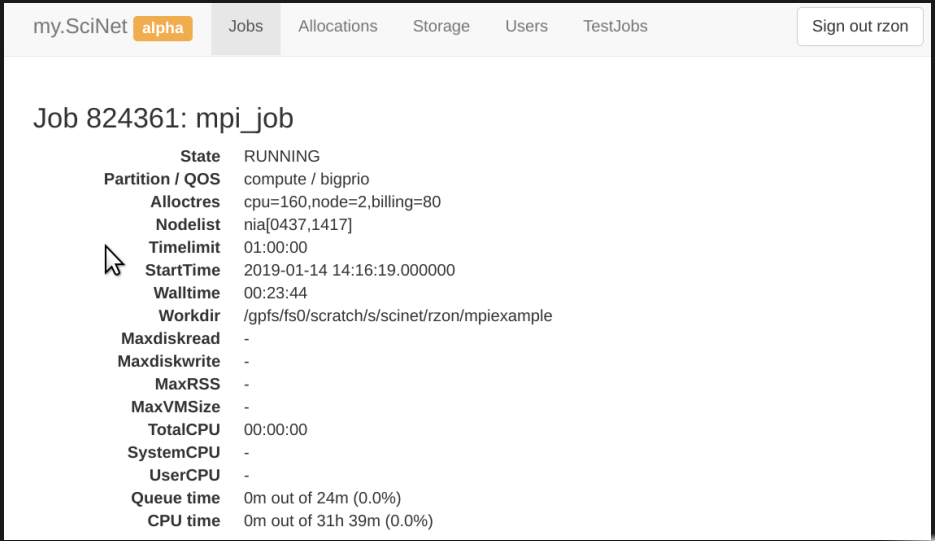
Check out https://my.scinet.utoronto.ca for past and present job info.



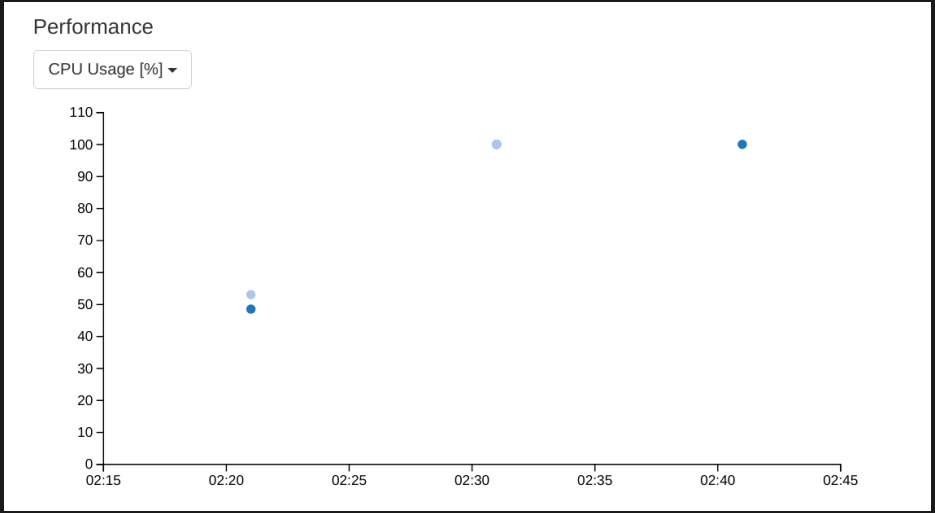
Features

- Niagara CPU and storage utilization
- Status of the login nodes
- Niagara and Mist job history
- Per job:
  - jobscript
  - environment
  - wall time
  - memory usage every 10 minutes.
  - CPU usage every 10 minutes.
  - GFlops/s every 10 minutes.
  - disk I/O usage every 10 minutes.

Usage: my.scinet.utoronto.ca



Usage: my.scinet.utoronto.ca



Usage: my.scinet.utoronto.ca

Script

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks=80
#SBATCH --time=1:00:00
#SBATCH --job-name mpi_job
#SBATCH --output=mpi_output_%j.txt
#SBATCH --mail-type=FAIL

module load intel/2018.2
module load openmpi/3.1.0

mpirun ./mpi_example
```

Environment

```
SLURM_ACCOUNT=scinet
```

Usage: Hyperthreading

- Hyperthreading is a technology that leverages more of the physical hardware by pretending there are more logical cores than real ones.
- On Niagara, each physical core becomes 2 virtual cores, so nodes seem to have 80 cores.
- On Mist, each physical core becomes 4 virtual cores, so nodes appear to have 128 cores.

Using hyperthreading on Niagara

- First, ask for a certain number of nodes for your jobs and set `--ntasks-per-node=40`.
- This way you get to use all cores on the nodes, but without hyperthreading.

  (mpirun, srun, and the OS will automaticallly spread processes over the real cores)
- Then **test** if running 80xN MPI processes or threads gives you any speedup by setting `--ntasks-per-node=80`.

  Even when doing so, your usage will be counted ("billing") as 40xNx(walltime in years).

Information links

Useful sites

- Niagara: https://docs.alliancecan.ca/wiki/Niagara_Quickstart
- Mist: https://docs.scinet.utoronto.ca/index.php/Mist
- Other Alliance clusters or general topics: https://docs.alliancecan.ca
- System Status: https://docs.scinet.utoronto.ca
- Training: https://education.scinet.utoronto.ca/

Support

- Email to **niagara@tech.alliancecan.ca** or **support@scinet.utoronto.ca**
- Still need help? Request a one-to-one consultation (request via email).