Ontario HPC Summer School 2016 — Central

Welcome to the 2016 Ontario Summer School on High Performance Computing in Toronto!

In this package you will find a schedule, login information, a SciNet quick reference guide for most of the hands-on sessions, and a command-line cheat sheet.

- **The location has changed to the Wallberg Building (see next page)!**
- **Bring a laptop with an ssh client and an X11 server**

  For Linux users, this is usually automatic; users of newer versions of OS X may need to install an external package such as such XQuartz. For Windows users, we ask you to install MobaXterm, from http://mobaxterm.mobatek.net. MobaXterm is a self-contained terminal/linux-like environment/X11-server installation that we at SciNet recommend to all our Windows users.
- **You need Wireless access for almost all sessions.** There are three options:

  1. UofT wireless network using your UTorID (Univ. of Toronto attendees); http://help.ic.utoronto.ca/content/20/704/en/wireless-access.html
  2. Eduroam (attendees from other Universities); http://www.canarie.ca/en/caf/participants
  3. Temporary access (everyone else). Simply ask the organizers for a temporary wifi account, and hold on to that information during the week!
- **You need access to the SciNet "GPC" system, and, for the CUDA sessions, to SHARCNET's GPU system.**

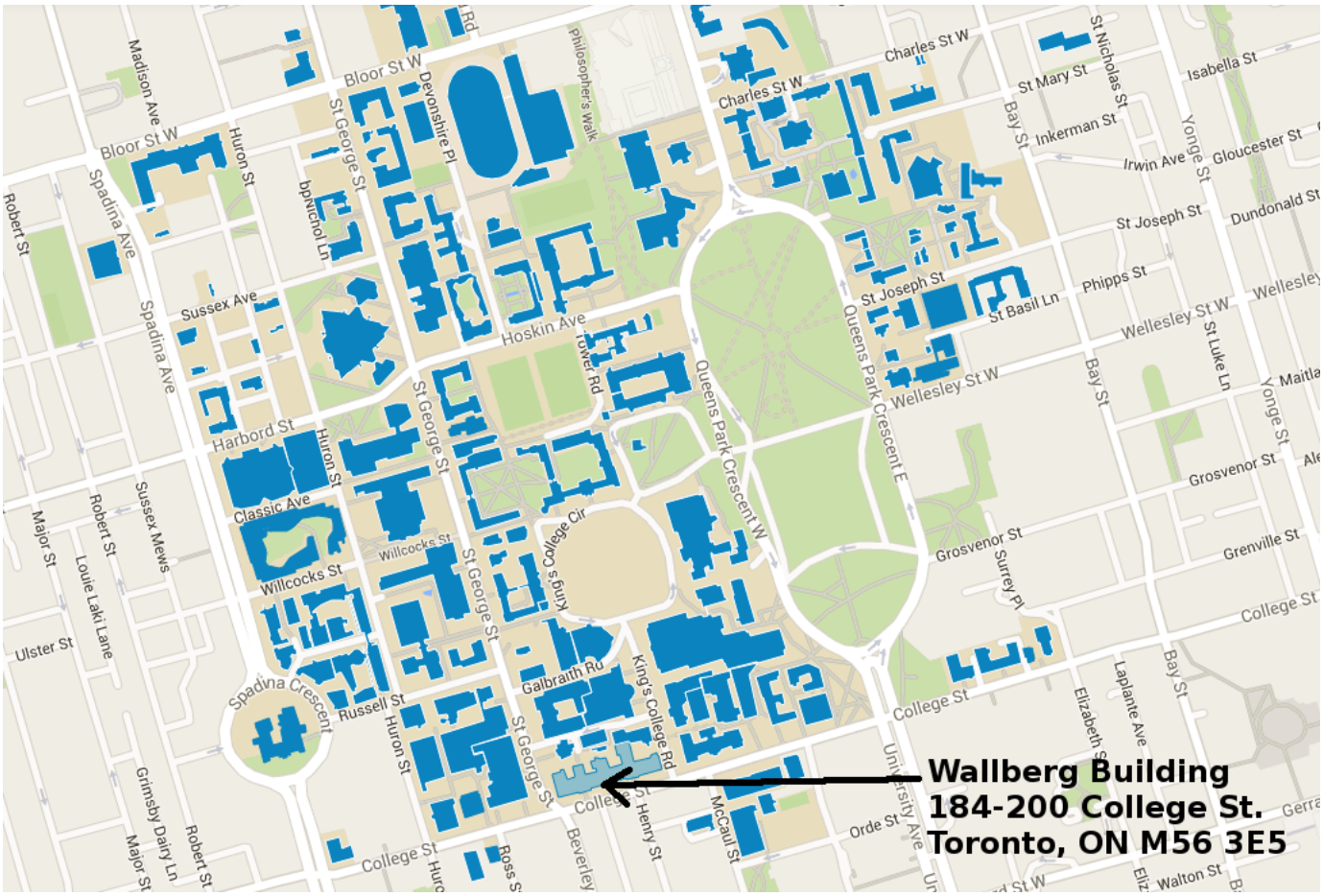
  If you do not have accounts, temporary accounts will be assigned to you. Remember to bring the temporary username and password with you during the week!
- **Registration will start around 9:15 am on Monday July 11, in Room 116 of the Wallberg Building.**
- **There will be short breaks around 11:00 am and 3:00 pm.**

  Coffee, tea and water, and a snack will be provided. Lunch breaks are from 12:30 to 13:30, however lunches are not provided; there are many options on and around campus.
- **A group photo will be taken on Wednesday afternoon at the start of the break.**
- **Participants who attend at least 3 days of the school get a summer school certificate.**
- **Course material will be available before the start of each session on the SciNet wiki:**

  http://tinyurl.com/jcnn2kg

New Location!

Wallberg Building
Rooms 116 and 119
University of Toronto
St. George Campus
184-200 College Street
Toronto, Ontario, M56 3E5



Schedule

Monday July 11	9:30 – 12:30	Intro to HPC and SciNet	WB 116
	12:30 – 13:30	Lunch break	
	9:30 – 16:30	OpenMP	WB 116
	9:30 – 12:30	Linux Shell	WB 119
Tuesday July 12	9:30 – 12:30	OpenMP	WB 116
	9:30 – 12:30	R for Data Science	WB 119
	12:30 – 13:30	Lunch break	
	13:30 – 16:30	MPI	WB 116
	13:30 – 16:30	Parallel R	WB 119
Wednesday July 13	9:30 – 12:30	MPI	WB 116
	9:30 – 12:30	Python for Science	WB 119
	12:30 – 13:30	Lunch break	
	13:30 – 16:30	MPI	WB 116
	13:30 – 16:30	HPC Python	WB 119
Thursday July 14	9:30 – 12:30	CUDA	WB 116
	9:30 – 12:30	Visualization	WB 119
	12:30 – 13:30	Lunch break	
	13:30 – 16:30	CUDA	WB 116
	13:30 – 16:30	Visualization	WB 119
Friday July 15	9:30 – 12:30	CUDA	WB 116
	9:30 – 12:30	Debugging	WB 119
	12:30 – 13:30	Lunch break	
	13:30 – 16:30	CUDA	WB 116
	13:30 – 16:30	Bring your own code	WB 119

Short breaks at 11 am and 3 pm.

Stream 1 is in room WB 116

Stream 2 is in room WB 119

SciNet Login Information

There will be a number of hands-on components in this summer school. For most sessions, we will be using SciNet's General Purpose Cluster (GPC). Here we'll show you how to log into the GPC and access the course material.

The CUDA sessions will use temporary SharcNet accounts to be given during those sessions. The Intro to Shell and the Visualization session will mostly use the participants' own laptop.

Your account

If you do not have a SciNet account, you will get a temporary SciNet username and password. For the CUDA session, you can get a temporary SHARCNET account.

Logging into the GPC

Once you have internet access, the first step is to login to SciNet's data centre login nodes. Open an Xwindows terminal (any terminal on a Linux Machine; under Mac, start Applications/Utilities/X11.app or Terminal.app; under Windows, start MobaXterm). From there

```
$ ssh -Y USERNAME@login.scinet.utoronto.ca
```

The `-Y` enables X-Windows forwarding (i.e., graphics).

These login nodes allow access to SciNet's clusters. From here, ssh to a development node on the GPC cluster

```
$ ssh -Y gpc
```

The devel nodes are shared among users. Since we're doing parallel programming, you'll want an entire (8-core) node to yourself, so request interactive access to a compute node (the first letter in the options is a lowercase L, the second to last is a capital I):

```
$ qsub -l nodes=1:ppn=8,walltime=7:00:00 -q teach -I -X
```

Course material

The source code used in the courses can be accessed by copying files into you directory.

```
$ cd $SCRATCH
$ cp -r /scinet/course/ss2016 .
```

The code is organized in directories, most of which contain a 'setup' script:

```
$ cd $SCRATCH/ss2016/.../code
$ source setup
```

This ensures we all have a consistent set of environment variables.

GPC Quick Start Guide

Logging In

SciNet allows login only via ssh, a secure protocol. Log into the login machines at the data centre, and then into the development nodes, where you do all your work. Batch computing jobs are run on the compute nodes.

For Linux/MacOS users

From a terminal window,

```
$ ssh -Y [USER]@login.scinet.utoronto.ca
scinet01-$ ssh -Y gpc01 (or gpc02, gpc03, gpc04.)
```

The first command logs you into the login nodes (replace `[USER]` with your username), the second logs you into one of the four development nodes. `-Y` allows Xwindows programs to pop up windows on your local machine.

For Windows users

For ssh we suggest:

The cygwin environment (`http://cygwin.com`), a linux-like environment. Be sure to install X11 and OpenSSH, and you can then (after launching the X11 client) run the commands listed above; or

MobaXterm (`mobaxterm.mobatek.net`), a tabbed ssh client.

Machine Details

Each GPC node has 8 processors, ∼14GB of free memory, and supports up to 16 threads or processes. Jobs are allocated entire nodes and must make full use of each.

Modules

Software is accessed by loading `modules` which place the package in your environment.

`module avail`	List packages.
`module load [pkg]`	Use default version of [pkg].
`module load [pkg]/[v.]`	Use version [v.] of [pkg].
`module unload [pkg]`	Remove [pkg] from path.
`module purge`	Remove all packages from path.

Common modules:

`module load gcc intel`	gcc, intel compilers.
`module load openmpi`	OpenMPI
`module load intelmpi`	Intel MPI (recommended)

Editors

Text-based editors are more responsive over a network connection than graphical editors, but both are available.

`vi filename`	vi editor.
`gvim filename`	vi editor (graphical)
`module load emacs`	
`emacs filename`	emacs editor (text).
`emacs -x filename`	emacs editor (graphical)
`module load nano`	
`nano filename`	Simple nano editor (text).

Disk

`/home/[USER]`	10 GB, backed up.
`/scratch/[USER]`	Large capacity; not backed up; purged every 3 months.
`module load extras`	Shows user, group disk usage
`diskUsage`	on /home, /scratch.

All SciNet nodes see the same filesystems. /home can only be *read* from on the compute nodes; batch jobs must be run from /scratch. The shared disk system is optimized for high bandwidth large reads and writes. Using many small files, or doing many small inputs and outputs, is inefficient and slows down the file system for all users.

Copying Files

`scp` copies files via the secure ssh protocol. Small (few GB) files may be copied to or from the login nodes. Eg, from your local machine, to copy files *from* SciNet,

```
scp -C [USER]@login.scinet.utoronto:~/[PathToFile]
  [LocalPathToNewFile]
```

copies [PathToMyfile] to the local directory. To copy a file *to* SciNet:

```
scp -C [myfile]
  [USER]@login.scinet.utoronto:~/[PathToNewFile]
```

Large files must be sent through datamover nodes; see the SciNet wiki for details.

Running Jobs

It is ok to run short (few minute), small-memory tests on the development nodes. Others must be run on the compute nodes via the queues, from the /scratch directory.

Debug queue

A small number of compute nodes are set aside for a debug queue, allowing short jobs (under 2 hours) to run quickly. To get a single debug node for an hour to run interactively,

```
qsub -I -l nodes=1:ppn=8,walltime=1:00:00 -q debug
```

and one can run as if one were logged into the devel nodes. One can also run short debug nodes in batch mode.

Batch queue

The usual usage of SciNet is to build and compile your code on /home, then copy the executable and data files to a directory on /scratch, write a script which descibes how to run the job, and submit it to the queue. When resources are free, your job runs to completion. Jobs in the batch queue may run no longer than 48 hours per session. Sample scripts follow.

Sample batch script - MPI

```
#!/bin/bash
#PBS -l nodes=2:ppn=8          Request 2 nodes
#PBS -l walltime=1:00:00       ..for 1 hour.
#PBS -N test                   Job name
cd $PBS_O_WORKDIR              cd to submission dir.
mpirun -np 16 [prog]           Run program w/ 16 tasks.
```

Sample batch script - OpenMP

```
#!/bin/bash
#PBS -l nodes=1:ppn=8          Request 1 node
#PBS -l walltime=1:00:00       ..for 1 hour.
#PBS -N test                   Job name
cd $PBS_O_WORKDIR              cd to submission dir.
export OMP_NUM_THREADS=8       Run with 8 OpenMP threads
[prog] > job.out               Run, save output in job.out.
```

Sample batch script - Serial Jobs

It is also possible to run batches of 8 serial jobs on a node to make sure the node is fully utilized. If all tasks take roughly the same amount of time:

```
#!/bin/bash
#PBS -l nodes=1:ppn=8          Request 1 node
#PBS -l walltime=1:00:00       ..for 1 hour.
#PBS -N serialx8               Job name
cd $PBS_O_WORKDIR              cd to submission dir.
(cd jobdir1; ./dojob1) &       Start task 1
(cd jobdir2; ./dojob2) &
...
(cd jobdir8; ./dojob8) &
wait                           Wait for all to finish
```

For more complicated cases, see the wiki.

Queue Commands

`qsub [script]`	Submit job to batch queue
`qsub [script] -q debug`	Submit job to debug queue
`qstat`	Show your queued jobs
`showq --noblock`	Show all jobs
`checkjob [jobid]`	Details of your job [jobid]
`showstart [jobid]`	*Estimate* start time
`canceljob [jobid]`	Cancel your job [jobid]

Ramdisk

Some of a node's memory may be used as a "ramdisk", a very fast filesystem visible only on-node. If your job uses little memory but does many small disk inputs/outputs, using ramdisk can significantly speed your job. To use: Copy your inputs to /dev/shm; cd to /dev/shm and run your job; then copy outputs from /dev/shm to /scratch.

Other Resources

`http://wiki.scinet.utoronto.ca`	Documentation
`support@scinet.utoronto.ca`	Email us for help

Shell-command cheat sheet

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`mkdir dir`	create a directory
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`cmd >> file`	append output to file
`cmd < file`	use file as input to cmd
`head file`	print first 10 lines of file
`tail file`	print last 10 lines of file
`wc file`	word count data of file
`logout`	close the terminal session

`wget url`	downloads the url
`tar file`	handles tar files
`cmd1 | cmd2`	pipe cmd1 output to cmd2
`sort file`	sorts the lines of file
`source file`	run the cmds in file
`grep arg file`	search for arg in file
`cut flags output`	cut part of output
`for..do..done`	for loop in bash
`if..then..fi`	if statement in bash
`arg`	mandatory argument
`[arg]`	optional argument

SciNet