Objectives

WestGrid and other campus facilities:

- Overview
- How to access
- How to use

HPC in Canada

In January 2005, a national long-range plan (LRP) for HPC across Canada was proposed by `c3.ca`, which was at the time the national advocacy group for HPC.

This plan envisioned creating a sustained, world-class, physical and human infrastructure for computation-based research.

In July 2005, the Canadian Foundation for Innovation (CFI) announced a National Platforms Fund competition to fund the LRP for HPC.

In December 2006, CFI announced $60M over 3 years for HPC equipment to the newly formed *Compute/Calcul Canada* (plus $18M in infrastructure operating and $10M from NSERC for personnel).

Compute Canada is now the over-arching governance structure for shared HPC infrastructure in Canada.

`c3.ca` disbanded in late 2007.

Overview of WestGrid

Before Compute Canada, shared HPC infrastructure in Canada was divided among 7 regional consortia: WestGrid, SHARCNET, SciNet, HPCVL, RQCHP, CLUMEQ, and AceNet.

Now there are four regional divisions: Compute West (WestGrid), Compute Ontario, Calcul Québec, and Compute Atlantic (ACEnet).

WestGrid presently consists of 14 partner institutions across the provinces of BC/AB/SK/MB.

Of these, there are 7 major partners (UVic, UBC, SFU, UofA, UofC, UofS, and UofM) that host the various pieces of shared infrastructure.

The other partners are UNBC, Lethbridge, Athabasca, the Banff Centre, UofR, Winnipeg, and Brandon.

Most (if not all) partners across Canada have AccessGrid Nodes for collaboration, and most have advanced visualization capabilities.

The shared infrastructure is mainly for supercomputing; Compute Canada is well-networked so that users can access resources regardless of where they are located.

The types of supercomputing facilities are (commodity) clusters, clusters with fast interconnect, and shared-memory systems.

There is also a number of GPU-based systems.

The UofS hosts a data storage centre, officially the largest in Compute Canada, with 3.15 PB of disk storage and 2.3 PB of tape storage.

WestGrid also has some licenses for popular software packages such as MATLAB, Gaussian (chemistry), OpenFOAM (CFD), and BLAST (bioinformatics).

Compute Canada offers support for collaboration, visualization, data transfer, and program optimization.

UofS HPC Training Clusters

The shared national infrastructure is a fantastic resource for running HPC jobs, but it is not well-suited for training or code development.

To aid UofS researchers make use of the national HPC resources as well as to complement individual research clusters, ITS has made available four machines for

- training of HQP in theory and implementation of parallel programming and parallel programs
- parallel code development / testing / debugging for research (called “staging”)

These machines are not intended to replace researcher clusters or Compute Canada resources.

Access to the training cluster machines is by virtue of your enrollment in this course; login is with your nsid.

Machines are behind the university firewall and so access must appear to be from an on-campus machine.

UofS Compute Cluster (`socrates`)

In May 2009, ITS commissioned a 37-node HPC cluster named `socrates` that has

- 1 head node (`socrates.usask.ca`)
- 8 capability nodes (`compute0-0` to `compute0-7`)
- 28 capacity nodes (`compute0-8` to `compute0-35`)

The designated use for `socrates` is for distributed-memory programs (1 Gigabit Ethernet interconnect).

Compilers available are `gcc`, `g77`, `gfortran`, `ifort`, and `icc` as well as the wrappers `mpicc` and `mpif77`.

The operating system is RedHat Enterprise Linux 5.3 with OSCAR clustering software.

Matlab and Mathematica are also available.

Jobs are submitted through a batching system (TORQUE/Maui).

UofS Large-Memory System *(*`moneta`*)*

In September 2009, ITS commissioned a large-memory machine called `moneta` that has

- 4 Intel Xeon E7430 quad-core processors (16 cores),
- 256 GB RAM,
- 64-bit RedHat Enterprise Linux 5.4,
- 500 GB of scratch disk for storing intermediate data.

The designated use of `moneta` is for large shared-memory programs.

Compilers available are `gcc`, `g77`, and `gfortran`, all available in `/usr/bin`.

Software available includes MATLAB, Mathematica, Maple, and R.

UofS tightly coupled, GPU-Accelerated System (`zeno`)

In September 2012, ITS commissioned a tightly coupled, GPU-accelerated machine called `zeno` with

- 8 nodes (2 Intel Xeon E5649 hex-cores and Tesla M2075 6 GB GPU (515 GFlops peak double precision, 1 TFlop peak single precision; 448 cores; memory bandwidth 150 GBytes/s))
- 24 GB RAM; 120 GB SATA HD
- high-speed InfiniBand interconnect

The goal of this machine is to facilitate training and experience with "tightly coupled systems" (a computing paradigm somewhere between the classical shared and distributed memory paradigms) and the ever-increasingly popular GPU-accelerated computing.

Cuda 4.2 and OpenCL are available.

UofS Research Cluster (plato)

In May 2013, ITS commissioned a 33-node cluster named plato that has

- 1 head node (plato.usask.ca): 2 8-core Intel Xeon processors; 32 GB RAM, 4TB RAID

- 32 computational nodes (compute0-0 to compute0-31): 2 8-core Intel Xeon E5-2650L processors; 32 GB RAM; local HD for scratch

- 1 Gb Ethernet between nodes; 10 Gb Ethernet between head node and private network

- Centos 6.3 Linux / ROCKS clustering software

Compilers available are gcc, g77, gfortran, ifort, and icc.

Generally not available for instructional purposes.

UofS WestGrid Collaboration and Visualization Facility (AG 2D71)

In September 2009, ITS commissioned a facility is designed to support advanced visualization and remote research collaborations.

Collaboration technologies include SmartBoard, AccessGrid, LifeSize Room 200 for H.323 videoconferencing and teleconferencing.

Facility allows for effective collaboration between a few researchers or 20+ people for remote presentations, e.g., WestGrid and Coast2Coast Seminar Series.

A CyViz Viz3D stereo optical projector system enables stereo 3D visualization of data. Remote visualization from other institutions is supported.

Equipment to support this collaboration technology includes multiple dedicated servers, 4 video cameras, 3 high-resolution projectors, an 19-foot custom screen, echo cancellation audio system, and wireless microphones and speaker phone.

How to access WestGrid

You are eligible to have a WestGrid account if you are associated with an eligible Canadian research project.

In general, any academic researcher from a Canadian research institution with significant HPC research requirements may apply for an account on WestGrid. A project description is required.

Students require sponsorship from a faculty supervisor, i.e., by joining an approved project.

There is a single point from which requests for accounts are generated and approved (see the WestGrid website).

Identical accounts are then "automatically" created on the various WestGrid clusters.

Once an account is created, users can then login, transfer files, etc. to any WestGrid machine using a standard protocol such as `ssh` as they would with any other UNIX workstation.

How to use WestGrid

Each major partner in WestGrid has a *Site Lead*, a technically oriented person who oversees the operation and maintenance of the shared infrastructure and provides a local point of contact to WestGrid.

At the UofS, the WestGrid Site Lead is Jason Hlady.

Jason is available to help with anything from finding more about WestGrid resources to setting up and using and account to helping your programs run more efficiently (or at all!) on WestGrid.

`jason.hlady@usask.ca`

All WestGrid computers use a UNIX variant or Linux operating system.

As mentioned, work such as job preparation, editing, compiling, testing, and debugging code may be done interactively on WestGrid machines, but this is not a recommended practice; use the UofS HPC training resources `socrates` and `moneta` instead.

The majority of the WestGrid computing resources are available only for batch-oriented production computing.

In other words, users must use a UNIX shell scripting language to write *job scripts* to run their programs.

Job scripts are submitted to the batch-job handling system (or queue) for assignment to a machine. The results are reported to the user upon job completion.

There is often a significant time lag between job submission and assignment, so this is an extremely inefficient way to (for example) debug code.

Every user is given a default allocations to WestGrid resources (access to CPUs and disk space).

An active user not requiring large memory or processor requirements would have access to 20–80 processors (depending on the machine) on a fairly regular basis.

Researchers desiring more than their default allocation for their work must submit a request for more resources to the *Resource Allocation Committee* (RAC).

Requests are measured in terms of *CPU-years*.

Running batch jobs

The system software that handles batch jobs consists of two pieces: a resource manager (TORQUE) and a scheduler (Moab).

Batch job scripts are UNIX shell scripts (basically text files of commands for the UNIX shell to interpret, similar to what you could execute by typing directly at a keyboard) containing special comment lines that contain *TORQUE directives*.

TORQUE evolved from software called Portable Batch System (PBS).

So TORQUE directive lines begin with #PBS, some environment variables contain “PBS”, and the script files typically have a .pbs suffix (although not required).

Note: There are small differences in the batch job scripts, particularly for parallel jobs, among the various WestGrid systems! See specific instructions for individual machines on WestGrid website.

Example: Job script `diffuse.pbs` for a serial job on glacier to run a program named `diffuse`.

```
#!/bin/bash
#PBS -S /bin/bash

# Script for running serial program, diffuse, on glacier

cd $PBS_O_WORKDIR
echo "Current working directory is ‘pwd‘"

echo "Starting run at: ‘date‘"
./diffuse
echo "Job finished with exit code $? at: ‘date‘"
```

To submit the script `diffuse.pbs` to the batch job handling system, use the `qsub` command:

```
qsub diffuse.pbs
```

If a job will require more than the default memory or time (typically 3 hours) allocation, additional arguments may be added to the `qsub` command.

If `diffuse` is a parallel program, the number of nodes on which it is to run must be specified, e.g.,

```
qsub -l walltime=72:00:00,mem=1500mb,nodes=4 diffuse.pbs
```

When `qsub` processes the job, it assigns it a `jobid` and places the job in a queue to await execution.

The status of all the jobs on the system can be displayed using

```
showq
```

To show just the jobs associated with your user name, use

```
showq -u username
```

To delete a job from the queue (or kill a running job), use `qdel` with the `jobid` assigned from qsub:

```
qdel jobid
```

It is wise for programs to periodically save output to a file so you can see how they are doing (and restart from that point if necessary). This is called checkpointing.

Sometimes, e.g., if you need to confirm how much memory your job is using, you may have to send e-mail to `support@westgrid.ca` to request that an administrator check on the job for you.

Other useful commands: <command> *job.id*

`qstat`: examine the status of a job

`qalter`: alter a job (specify attributes)

`qhold`: put a job on hold

`qorder`: exchange order of two jobs (specify jobids)

`qrls`: release hold on a job

`qsig`: send a signal to a job (specify signal)

See also

`http://www.clusterresources.com/torquedocs21/`

Other PBS directives

```
# Set the name of the job (up to 15 characters,
# no blank spaces, start with alphanumeric character)
#PBS -N JobName

# make pbs interpret the script as a bash script
#PBS -S /bin/bash

# specify filenames for standard output and error streams
# By default, standard output and error streams are sent
# to files in the current working directory with names:
#     job_name.osequence_number  <-  output stream
#     job_name.esequence_number  <-  error stream
# where job_name is the name of the job and sequence_number
# is the job number assigned when the job is submitted.

#PBS -o stdout_file
#PBS -e stderr_file
```

```
# Specify the maximum cpu and wall clock time.
# cput  =
# walltime =
# The wall clock time should take queue waiting time into
# account.  Format:   hhhh:mm:ss   hours:minutes:seconds
# Be sure to specify a reasonable value here.
# If the job does not finish by the time reached,
# the job is terminated.

#PBS -l     cput=2:00:00
#PBS -l walltime=6:00:00

# Specify the maximum amount of physical memory required.
# kb for kilobytes, mb for megabytes, gb for gigabytes.
# mem  = max amount of physical memory used by all processes
# Take care in setting this value.  Setting it too large
# can result in the job waiting in the queue for sufficient
# resources to become available.

#PBS -l mem=512mb
```

```
# PBS can send email messages to you about the
# status of your job.  Specify a string of
# either the single character "n" (no mail), or one or more
# of the characters "a" (send mail when job is aborted),
# "b" (send mail when job begins), and "e" (send mail when
# job terminates).  The default is "a" if not specified.
# You should also specify the email address to which the
# message should be send via the -M option.

#PBS -m abe
#PBS -M user_email_address

# Specify the number of nodes requested and the
# number of processors per node.

#PBS -l nodes=1:ppn=1

# Define the interval when job will be checkpointed
# in terms of an integer number of minutes of CPU time.

#PBS -c c=2
```

There is further help available for using PBS on WestGrid via the command `man pbs`.

User responsibilities

WestGrid is a *shared production HPC environment*.

This means that WestGrid is not good for developing code or learning how to use software.

Although some support is available, in practice users should learn enough UNIX to know how to transfer files, submit and monitor batch jobs, monitor disk usage, etc.

Users are expected to use the WestGrid systems responsibly!

Users should be able to estimate memory requirements (both RAM and disk) and run times for their jobs.

Code is expected to be optimized through appropriate choice of algorithm, compiler flags, and/or optimized numerical libraries.

How to use the UofS HPC resources

You should have accounts on `socrates`, `moneta`, and `zeno` by virtue of being enrolled in this course.

For security reasons, both machines are on the UofS private network and so cannot be accessed directly from off campus; i.e., users can only connect to these machines from another machine in the `usask` domain.

Login is done using your UofS NSID and password, e.g.,

```
ssh abc123@socrates.usask.ca
```

MATLAB is available on both machines and can be accessed simply by typing

```
matlab
```

Help is available by e-mailing `hpc_consult@usask.ca`.

Summary

- The HPC landscape in Canada and at the UofS
- Using HPC resources: from theory to practice