University of Manitoba

Digital Research Alliance of Canada

High Performance Computing: Start Guide

How to use Grex and get more from the available resources?

UofM-Autumn-Workshop 2022

Oct 26th-27th, 2022

Ali Kerrache

University of Manitoba

Outline

➔ Available resources for UofM researchers/collaborators:

- The Alliance: cedar, graham, beluga, narval, niagara, cloud.
- Grex: a local HPC clusters at **UofM.**

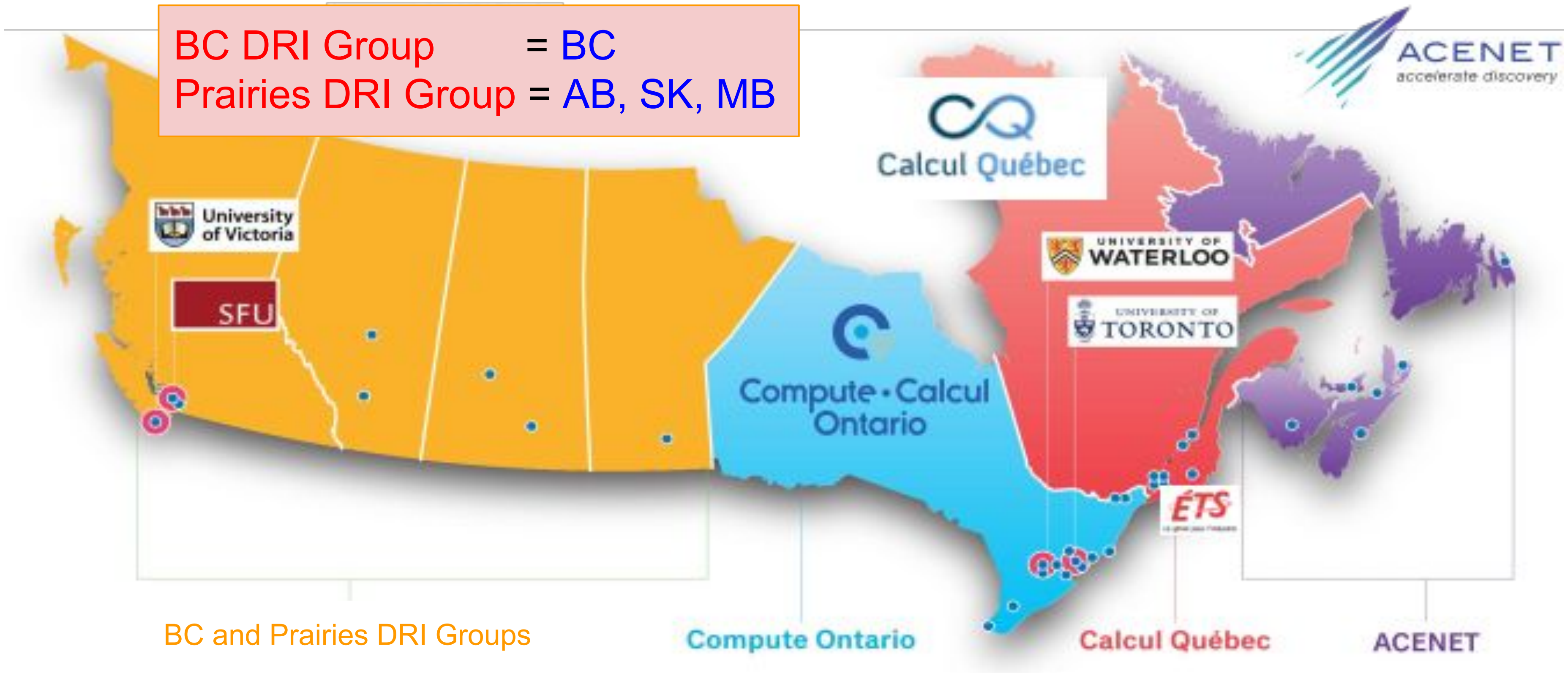
➔ Start Guide for using HPC resources:

- Get an account (+active role): CCDB
- Linux shell (Terminal, command line, edit files, …)
- Connect to a cluster: ssh, PuTTY, MobaXterm, X2Go, OOD
- Transfer files: scp, rsync, sftp, WinSCP, FileZilla, …
- Install programs and/or use existing modules (Lmod)
- Submit and monitor jobs: sbatch, salloc, squeue, seff … etc.

University of Manitoba

The Alliance clusters

BC DRI Group = BC
Prairies DRI Group = AB, SK, MB



University of Manitoba

The Alliance clusters

Cluster	Cores	GPUs	Storage	Notes
Cedar	94,528	1352	29 PB	NVidia P100; V100 Volta GPUs
Graham	41,548	520	19 PB	NVidia P100; V100; T4 GPUs
Beluga	28,000	688	27 PB	NVidia V100 GPUs
Narval	73,088	636	24.5 PB	NVidia A100 GPUs [40 GB memory]
Niagara; Mist	80,640	216	16 PB	Large parallel jobs; [4 NVIDIA V100-32GB]
Arbutus	16,008	108	17.3 PB	Physical cores: generally hyper-threaded.
GP cloud	-	-	-	Available on all General Purpose clusters.

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

University of Manitoba

Resources on Grex: partitions

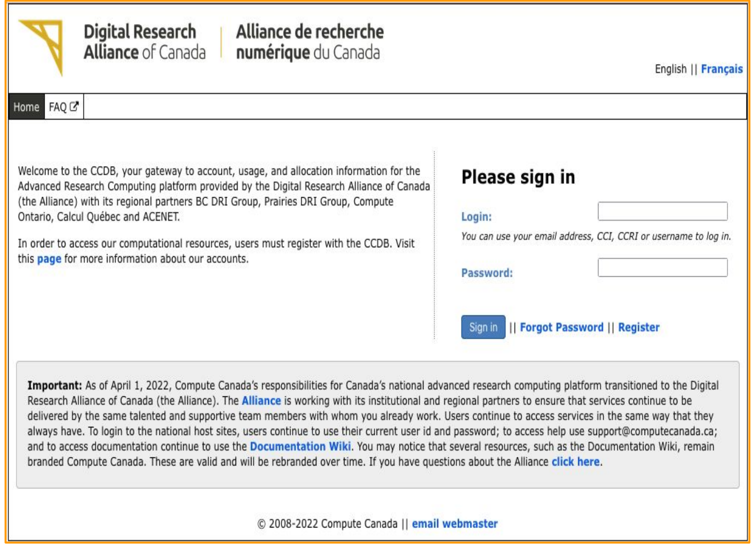
Partition	Nodes [CPUs/GPUs]	Cores	Total	Memory	Wall Time
compute[1]	312	12	3456	46 GB	21 days
largemem	12	40	480	376 GB	14 days
skylake	42	52	2184	96 GB	21 days
gpu	2 [4 V100 - 32 GB]	32	64	187 GB	3 days
stamps; -b	3 [4 V100 - 16 GB]	32	96	187 GB	21 days / 7 days
livi; -b	[16 V100 - 32 GB]	48	48	1.5 TB	21 days / 7 days
agro; -b	2 AMD [A30]	24	48	250 GB	21 days / 7 days
test	-	18	18	500 GB	12 hours

[1] to be decommissioned in the near future.

https://um-grex.github.io/grex-docs/

University of Manitoba

Access to Alliance clusters / Grex


Digital Research Alliance of Canada | Alliance de recherche numérique du Canada

English || Français

Welcome to the CCDB, your gateway to account, usage, and allocation information for the Advanced Research Computing platform provided by the Digital Research Alliance of Canada (the Alliance) with its regional partners BC DRI Group, Prairies DRI Group, Compute Ontario, Calcul Québec and ACENET.

In order to access our computational resources, users must register with the CCDB. Visit this page for more information about our accounts.

Please sign in

Login:

You can use your email address, CCI, CCRI or username to log in.

Password:

Sign in || Forgot Password || Register

Important: As of April 1, 2022, Compute Canada's responsibilities for Canada's national advanced research computing platform transitioned to the Digital Research Alliance of Canada (the Alliance). The Alliance is working with its institutional and regional partners to ensure that services continue to be delivered by the same talented and supportive team members with whom you already work. Users continue to access services in the same way that they always have. To login to the national host sites, users continue to use their current user id and password; to access help use support@computecanada.ca; and to access documentation continue to use the Documentation Wiki. You may notice that several resources, such as the Documentation Wiki, remain branded Compute Canada. These are valid and will be rebranded over time. If you have questions about the Alliance click here.

© 2008-2022 Compute Canada || email webmaster

Step 1: Principal Investigator (PI) or sponsor

Faculty member registers in the Alliance Database (CCDB): http://ccdb.computecanada.ca

Step 2: sponsored users

Once PI's account is approved, students / colleagues can register as group members (require CCRI).

The Alliance: Rapid Access Service

10 TB of storage / cluster.

Send an email to: support@tech.alliancecan.ca

RAC for storage > 10 TB.

CCDB account: gives access to new systems / Grex

- Access to resources is free for eligible researchers.
- Every group gets a "default" share; 1 TB of storage.
- Resource Allocation Competitions: about 80 %

  Held each year, valid for 1 year [April till end of March]
- Default Allocations: 20 % are used for default share.

Workflow on HPC clusters

Connect to a cluster

Linux:

ssh client, X2Go

Mac:

ssh client, X2Go

Windows:

Putty, MobaXterm

Transfer files

Linux, Mac:

scp, sftp, rsync

Windows:

WinScp, MobaXterm, FileZilla, PuTTy

HPC work

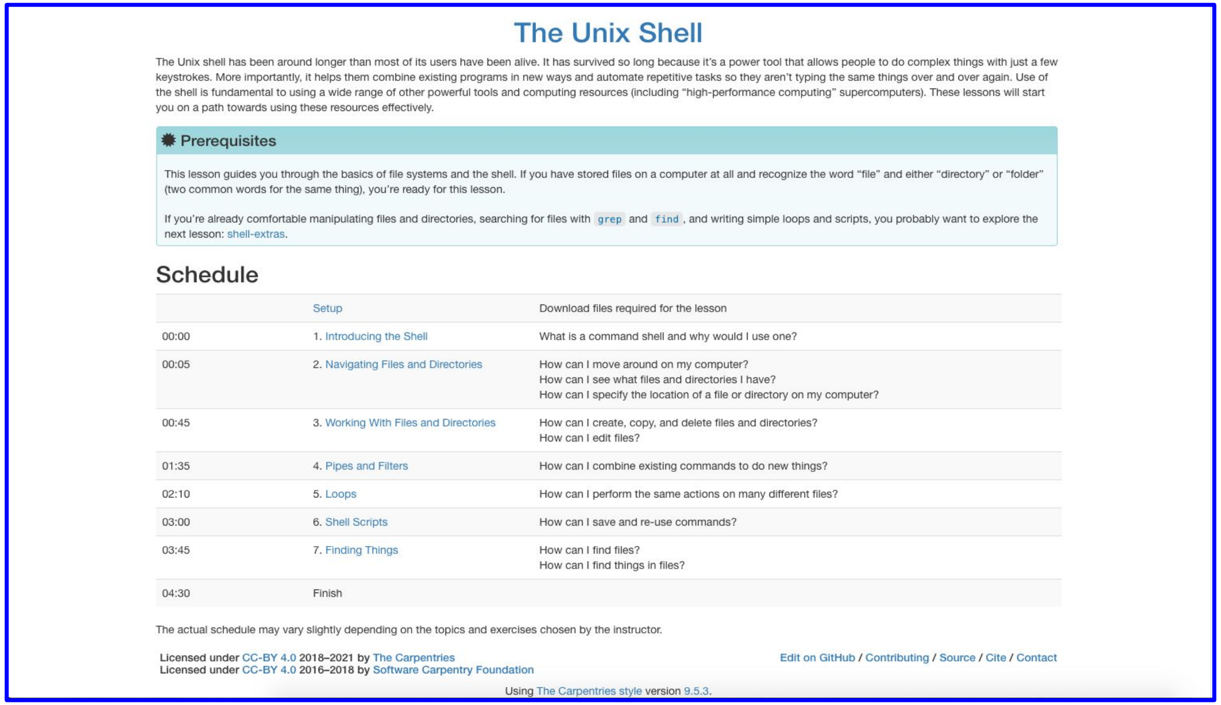
- Connect
- Transfer files
- Compile codes
- Test jobs
- Run jobs
- Analyze data
- Visualisation

OpenOnDemand: remote web access to supercomputers

https://um-grex.github.io/grex-docs/

University of Manitoba

Linux: carpentry courses



The Unix Shell

The Unix shell has been around longer than most of its users have been alive. It has survived so long because it's a power tool that allows people to do complex things with just a few keystrokes. More importantly, it helps them combine existing programs in new ways and automate repetitive tasks so they aren't typing the same things over and over again. Use of the shell is fundamental to using a wide range of other powerful tools and computing resources (including "high-performance computing" supercomputers). These lessons will start you on a path towards using these resources effectively.

Prerequisites

This lesson guides you through the basics of file systems and the shell. If you have stored files on a computer at all and recognize the word "file" and either "directory" or "folder" (two common words for the same thing), you're ready for this lesson.

If you're already comfortable manipulating files and directories, searching for files with `grep` and `find`, and writing simple loops and scripts, you probably want to explore the next lesson: shell-extras.

Schedule

	Setup	Download files required for the lesson
00:00	1. Introducing the Shell	What is a command shell and why would I use one?
00:05	2. Navigating Files and Directories	How can I move around on my computer? How can I see what files and directories I have? How can I specify the location of a file or directory on my computer?
00:45	3. Working With Files and Directories	How can I create, copy, and delete files and directories? How can I edit files?
01:35	4. Pipes and Filters	How can I combine existing commands to do new things?
02:10	5. Loops	How can I perform the same actions on many different files?
03:00	6. Shell Scripts	How can I save and re-use commands?
03:45	7. Finding Things	How can I find files? How can I find things in files?
04:30	Finish	

The actual schedule may vary slightly depending on the topics and exercises chosen by the instructor.

Carpentry courses for beginners:

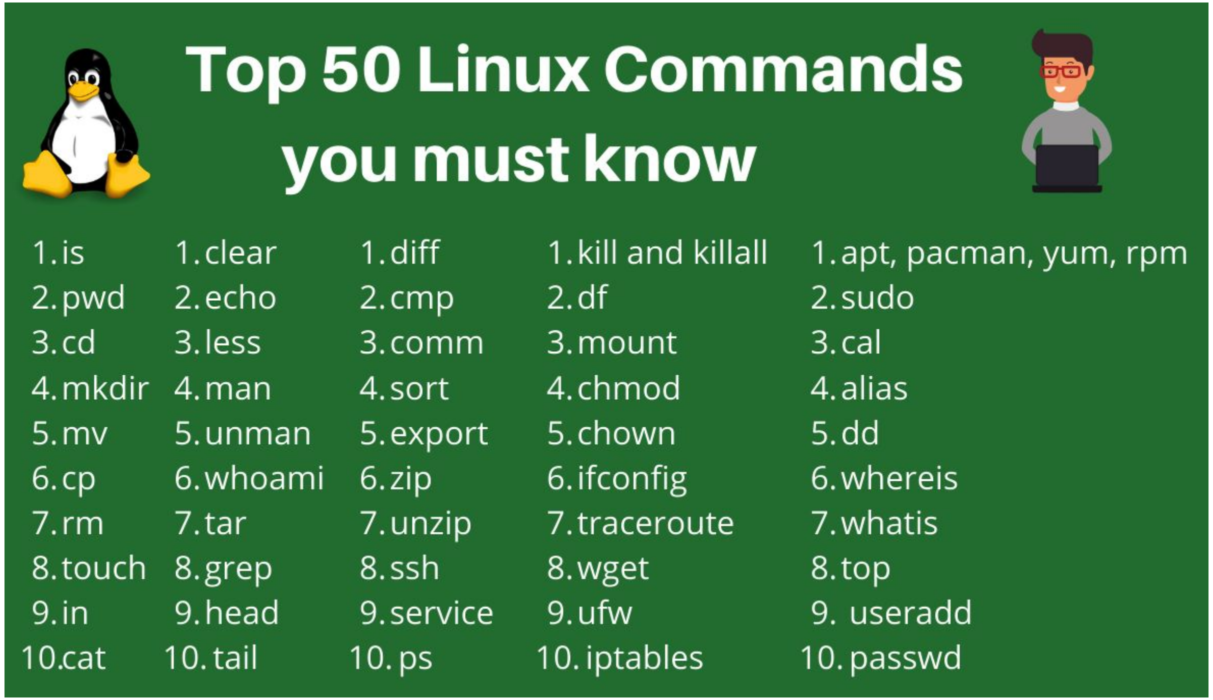
- Introducing the shell
- Navigating/working with files & directories
- Pipes and filters
- Loops
- Shell scripts
- Finding files/programs
- Automate tasks

https://swcarpentry.github.io/shell-novice/

https://training.westdri.ca/

University of Manitoba

Linux: basic commands



Top 50 Linux Commands you must know

1. is	1. clear	1. diff	1. kill and killall	1. apt, pacman, yum, rpm
2. pwd	2. echo	2. cmp	2. df	2. sudo
3. cd	3. less	3. comm	3. mount	3. cal
4. mkdir	4. man	4. sort	4. chmod	4. alias
5. mv	5. unman	5. export	5. chown	5. dd
6. cp	6. whoami	6. zip	6. ifconfig	6. whereis
7. rm	7. tar	7. unzip	7. traceroute	7. whatis
8. touch	8. grep	8. ssh	8. wget	8. top
9. in	9. head	9. service	9. ufw	9. useradd
10. cat	10. tail	10. ps	10. iptables	10. passwd

https://www.digitalocean.com/community/tutorials/linux-commands

Most used commands

- cd; mkdir; mv; rm; ls
- pwd;
- head, tail; less; more
- top; ps; htop
- gzip; tar; bzip2; gunzip
- zip; unzip
- wget; curl
- ssh; scp; sftp
- chmod; chgrp; …

man <command>

<command> –help; -h

Connect, transfer files, …

- ★ **ssh** **=>** Secure Shell
- ★ **scp** **=>** Secure Copy
- ★ **sftp** **=>** Secure File Transfer Protocol
- ★ **PuTTY** **=>** SSH and Telnet for Windows
- ★ **FileZilla** **=>** Utility for transferring files by FTP
- ★ **WinSCP** **=>** SFTP/FTP client for Microsoft Windows
- ★ **MobaXterm** **=>** Toolbox for remote computing
- ★ **X2Go** **=>** Remote desktop software for Linux
- ★ **OOD** **=>** Interface to remote computing resources

University of Manitoba

How to connect to a cluster?

Syntaxe:

```
~$ ssh [+options] <username>@<hostname>
```

options = -X; -Y {*X11 forwarding*}, …

➔ **Windows:** install PuTTy, MobaXterm, …

➔ **Mac:** install XQuartz {*X11 forwarding*}

Connect from a terminal:

```
Grex:   ~$ ssh -XY <username>@grex.hpc.umanitoba.ca
Grex:   ~$ ssh -XY <username>@yak.hpc.umanitoba.ca
Cedar:  ~$ ssh -XY <username>@cedar.computecanada.ca
Graham: ~$ ssh -XY <username>@graham.computecanada.ca
Beluga: ~$ ssh -XY <username>@beluga.computecanada.ca
Narval: ~$ ssh -XY <username>@narval.computecanada.ca
```

★ **password**

★ **ssh keys**

Very Important

Don't share your password with anyone.

Don't send your password by email.

In case you forgot your password, it is possible to **reset it** from **CCDB**.

University of Manitoba

Connect from Windows

- ❖ **Install ssh client:**
  - ➢ Putty: http://www.putty.org/
  - ➢ MobaXterm: https://mobaxterm.mobatek.net/
- ❖ **How to connect?**
  - ✓ **Login:** your user name
  - ✓ **Host:** grex.hpc.umanitoba.ca
  - ✓ **Password:** your password
  - ✓ **Port:** 22
- ❖ **Use CygWin:** same environment as Linux

Download PuTTY

Bitvise SSH Client

X2Go: Linux/Mac/Windows

Why X2Go: Access to GUI

How to use X2Go?

- Ask first if X2Go is installed on the remote machine.
- If yes, install X2Go client on your laptop or Desktop.
- Linux, Windows, Mac (XQuartz)
- Launch X2Go; Create a session and connect.

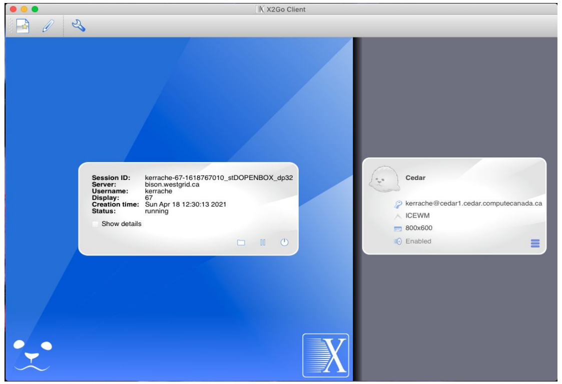
Login: your user name

Host: bison.hpc.umanitoba.ca

{or tatanka.hpc.umanitoba.ca}

Port: 22

Session: ICEWM

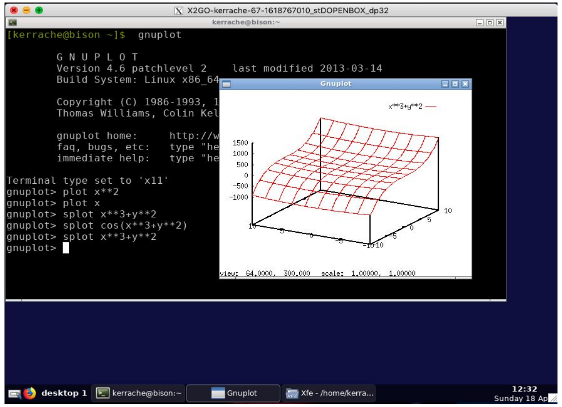


Session ID: kerrache-67-1618767010_stDOPENBOX_dp32
Server: bison.westgrid.ca
Username: kerrache
Display: 67
Creation time: Sun Apr 18 12:30:13 2021
Status: running

Show details

Cedar

kerrache@cedar1.cedar.computecanada.ca
ICEWM
800x600
Enabled

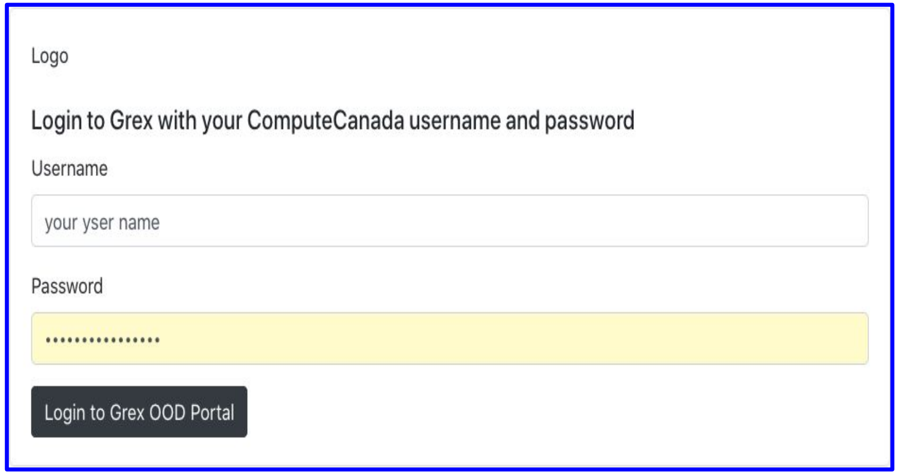


```
[kerrache@bison ~]$  gnuplot

        G N U P L O T
        Version 4.6 patchlevel 2    last modified 2013-03-14
        Build System: Linux x86_64

        Copyright (C) 1986-1993, 1
        Thomas Williams, Colin Kel

        gnuplot home:     http://w
        faq, bugs, etc:   type "he
        immediate help:   type "he

Terminal type set to 'x11'
gnuplot> plot x**2
gnuplot> plot x
gnuplot> splot x**3+y**2
gnuplot> splot cos(x**3+y**2)
gnuplot> splot x**3+y**2
gnuplot> 
```

```
view: 64.0000, 300.000   scale: 1.00000, 1.00000
```

OOD: OpenOnDemand web portal

Connect to OOD using: UManitoba VPN:
- ★ Make sure Pulse Secure VPN is connected
- ★ Point your Web browser to https://aurochs.hpc.umanitoba.ca
- ★ Use your Alliance (Compute Canada) username/password to log in to Grex OOD.



Logo

Login to Grex with your ComputeCanada username and password

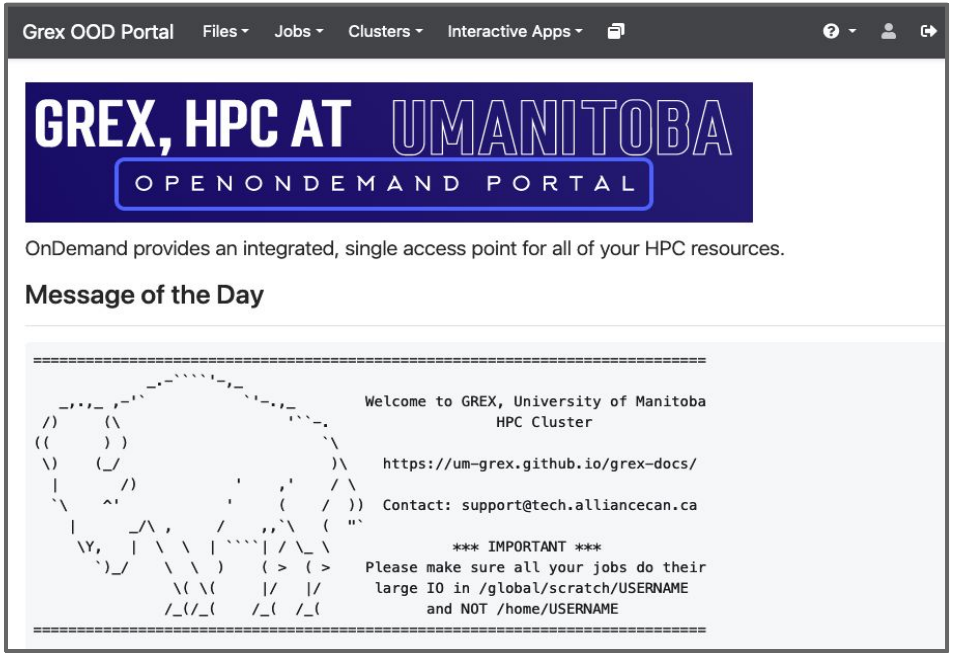
Username

your yser name

Password

••••••••••••••••

Login to Grex OOD Portal



OnDemand provides an integrated, single access point for all of your HPC resources.

Message of the Day

```
=============================================================================
             _.-````'-,_
   _,.,_ ,-'`           `'-.,_        Welcome to GREX, University of Manitoba
 /)     (\                   '``-.                   HPC Cluster
((      ) )                      `\
 \)    (_/                        )\    https://um-grex.github.io/grex-docs/
  |       /)           '    ,'    / \
  `\    ^'            '     (    /  ))  Contact: support@tech.alliancecan.ca
    |      _/\ ,     /    ,,`\   (  "`
     \Y,   |  \  \  | ````| / \_ \              *** IMPORTANT ***
       `)_/    \  \  )    ( >  ( >    Please make sure all your jobs do their
                \( \(     |/   |/      large IO in /global/scratch/USERNAME
               /_(/_(    /_(  /_(            and NOT /home/USERNAME
=============================================================================
```

- ★ Run jobs, View jobs, files, … etc.
- ★ Run MATLAB, Gaussview, Desktop, Jupyter, …

File system and quota

the Alliance [Compute Canada]:

/home/$USER: **50** GB, daily backup

/scratch/$USER: **20** TB, no backup, purged

Grex:

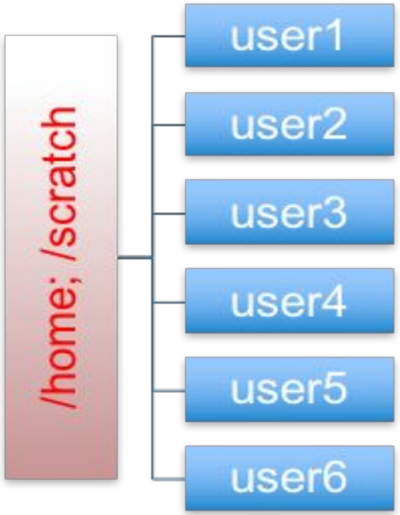
/home/$USER:

100 GB

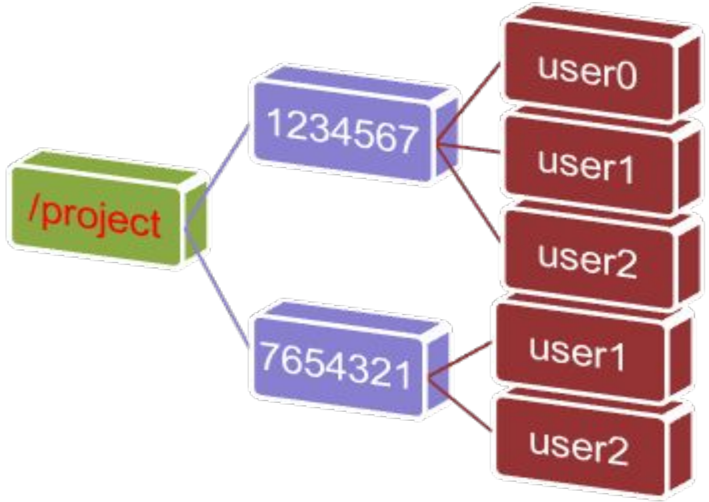
/global/scratch/$USER:

4 TB, no backup, no purge.

/project: no backup, no purge.



Project:



1 TB per group; extension up to 10 TB

Backup; Allocatable via RAC (>10 TB)

Quota: diskusage_report

```
[someuser@cedar1: ~]$ diskusage_report
                        Description           Space        # of files
            /home (user someuser)     →   50G/50G         6520/500k
         /scratch (user someuser)         12T/20T        8517/1000k
       /project (group someuser)          0/2048k            0/1025
   /project (group def-someprof)        1200G/10T     →   500k/500k
   /project (group rrg-someprof)        5838G/50T           250k/2M
```

Over quota

Space under home directory

Inode under project def-somep

```
[someuser@tatanka ~]$  diskusage_report
                 Description (FS)     Space (U/Q)     # of files (U/Q)
                 /home (someuser)      226M/104G        2381/500k
       /global/scratch (someuser)     519G/4294G        27k/1000k
```

University of Manitoba

File transfer: scp, sftp, rsync, ...

Terminal: Linux; Mac; CygWin; MobaXterm, PuTTy.

Check if scp; sftp; rsync are supported.

Syntax for scp: scp [+options] [Target] [Destination]

Syntax for rsync: rsync [+options] [Target] [Destination]

Options: for details use man scp or man rsync from your terminal.

Target: file(s) or directory(ies) to copy (exact path).

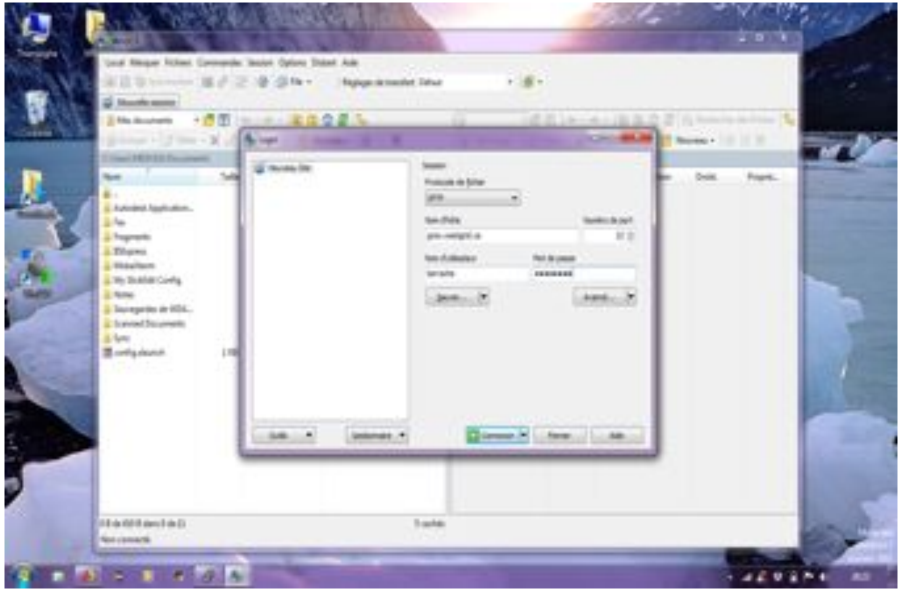
Destination: where to copy the files (exact path) [hostname:<full path>]

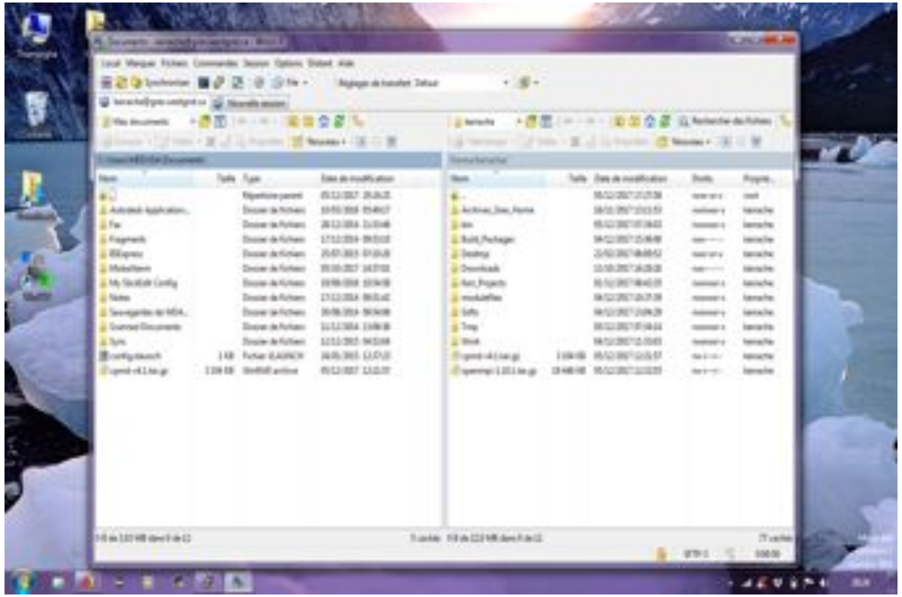
Path on remote machine: examples of a path on Grex.

```
username@grex.hpc.umanitoba.ca:/home/username/{Your_Dir}; ~/{Your_Dir}
username@grex.hpc.umanitoba.ca:/global/scratch/username/{Your_Dir}
[~@Mac]: scp -r TEST username@grex.hpc.umanitoba.ca:/global/scratch/username/Work
```

File transfer: FileZilla, WinSCP

- Install WinScp or FileZilla.
- Launch the program.
- Connect with your credentials.





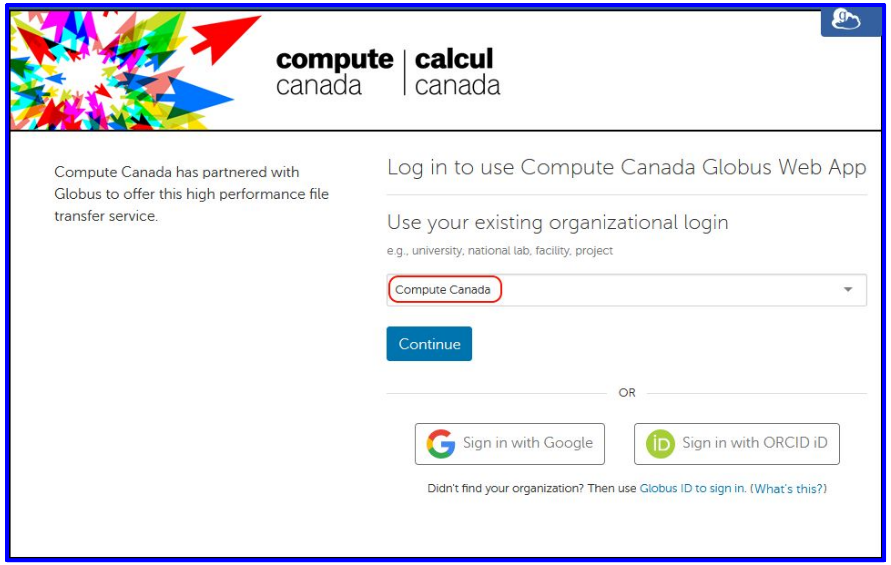
- Navigate on your local machine.
- Navigate on remote machine.
- Copy your files (works on both ways).

University of Manitoba

File transfer: Globus

- Launch Globus web interface.
- Connect with your credentials.

- Search for the globus endpoints
- Navigate to your directories
- Initiate the transfer / Log out.



compute canada | calcul canada

Compute Canada has partnered with Globus to offer this high performance file transfer service.

Log in to use Compute Canada Globus Web App

Use your existing organizational login

e.g., university, national lab, facility, project

Compute Canada

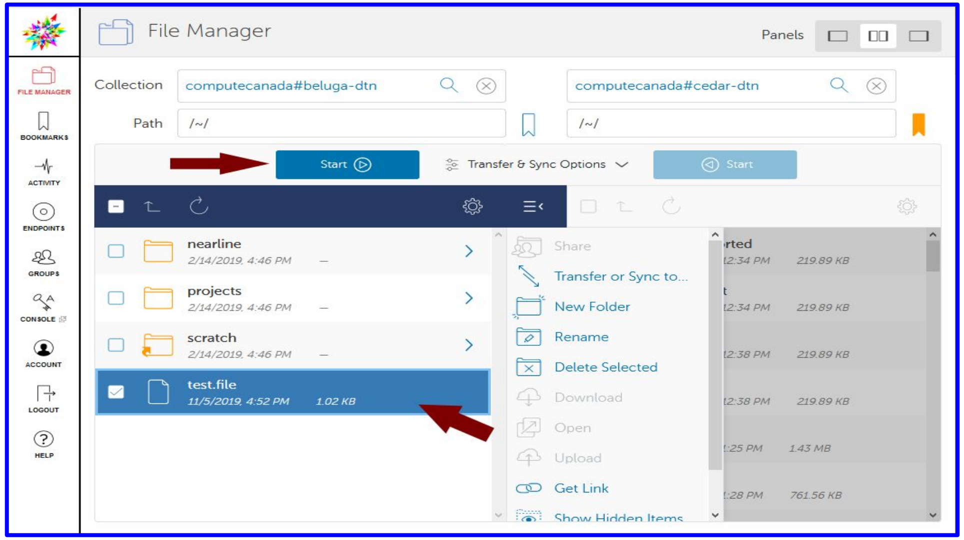
Continue

OR

Sign in with Google | Sign in with ORCID iD

Didn't find your organization? Then use Globus ID to sign in. (What's this?)

https://docs.alliancecan.ca/wiki/Globus/en



File Manager — Panels

Collection: computecanada#beluga-dtn | computecanada#cedar-dtn

Path: /~/ | /~/

Start | Transfer & Sync Options | Start

nearline 2/14/2019, 4:46 PM —
projects 2/14/2019, 4:46 PM —
scratch 2/14/2019, 4:46 PM —
test.file 11/5/2019, 4:52 PM 1.02 KB

Software and modules

- ★ Software on HPC clusters
- ★ Software distribution
- ★ Available software on HPC clusters
- ★ Find a software: modules

University of Manitoba

Software on HPC clusters

★ Home made: programs, scripts and tools, … etc.

Up to a user, …

★ Free Software: GNU Public License.

Open Source, Binaries, Libraries, Compilers, Tools, …

★ Commercial Software: restricted [VASP, STATA, …]

➔ Contact support with some details about the license, …

➔ We install the program and protect it with a POSIX group.

Software distribution

- Operating system package managers / repos
  - **Ubuntu:** $ ***sudo* apt-get install bowtie2**
  - **CentOS:** $ ***sudo* yum install bowtie2** # might need EPEL repo
  - **On HPC**, users do not have **sudo**! {**It is not required; no need to ask for it**}
- Local install from sources or binaries, usually to $HOME
  - wget https://github.com/BenLangmead/bowtie2/releases/download/v2.3.4.3/bowtie2-2.3.4.3-linux-x86_64.zip
  - unzip bowtie2-2.3.4.3-linux-x86_64.zip
  - bowtie2-2.3.4.3-linux-x86_64/bowtie2 -?
  - OR build from sources, specifying the PREFIX, **CMAKE_INSTALL_PREFIX** or **--prefix** to $HOME/bowtie2/
  - and add the locations to PATH, LD_LIBRARY_PATH etc.
- Using a centralized HPC stack
  - installed and maintained by analysts: compilers, libraries, domain specific software, … etc.
  - ask for installing a given program or updating modules if needed

Available software on HPC clusters

- ★ Number-crunching software environments:
  - ○ Compilers (GCC, Intel), BLAS/LAPACK/PETSc, BLIS, MPI, OpenMP, … etc.
- ★ Dynamic languages and libraries: R, Python, Perl, Julia, ...
- ★ Domain-specific applications and packages:
  - ○ Engineering, Chemistry, Physics, Machine-Learning, ...
  - ○ Biomolecular, genomics etc.
- ★ CC Centralized software stack, distributed via CVMFS:

  https://docs.alliancecan.ca/wiki/Available_software
- ★ **Grex:**
  - ○ **GrexEnv:** modules installed locally on Grex [more than 500 modules].
  - ○ **CCEnv:** access to public repository of the Alliance.

How to find a software?

★ **Why modules?**

- Control different versions of the same program.
- Avoid conflicts between different versions and libraries.
- Set the right path to each program or library.

★ **Useful commands for working with modules:**

- module **list**; module **avail**
- module **spider** <soft>/<version>
- module **load** soft/version; module **unload {rm}** <soft>/<version>
- module **show** soft/version; module **help** <soft>/<version>
- module **purge**; module --force **purge**
- module **use** ~/modulefiles; module **unuse** ~/modulefiles

```
[someuser@bison ]$  module list

Currently Loaded Modules:
  1) GrexEnv (S)

  Where:
   S:  Module is Sticky, requires --force to
unload or purge
```

Find and load QE

```
[someuser@bison ]$  module spider espresso

--------------------------------------------------------------------------------------------
  espresso:
--------------------------------------------------------------------------------------------

     Versions:
        espresso/5.4.0
        espresso/6.3
        espresso/6.4.1
        espresso/6.5
        espresso/7.0

--------------------------------------------------------------------------------------------
  For detailed information about a specific "espresso" package (including how to load the modules) use the
module's full name. Note that names that have a trailing (E) are extensions provided by other modules.
  For example:

     $ module spider espresso/7.0
--------------------------------------------------------------------------------------------
```

```
[someuser@bison ]$  module spider espresso/7.0
```

```
[someuser@bison ]$  module load intel/2020.4  ompi/4.1.0 espresso/7.0
   Loading module: hdf5-1.12.1
   Loading module: libxc/5.2.2
   Loading module: ESPRESSO/7.0
```

Find and load ORCA

ORCA:

- **restricted software**
- **requires a registration**

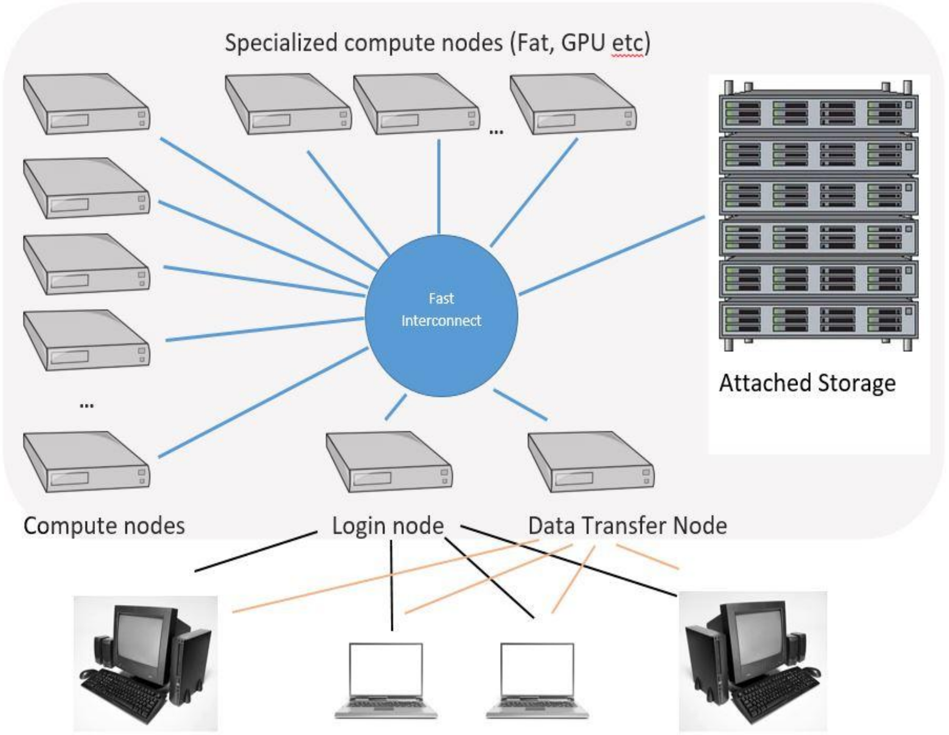
https://docs.alliancecan.ca/wiki/ORCA

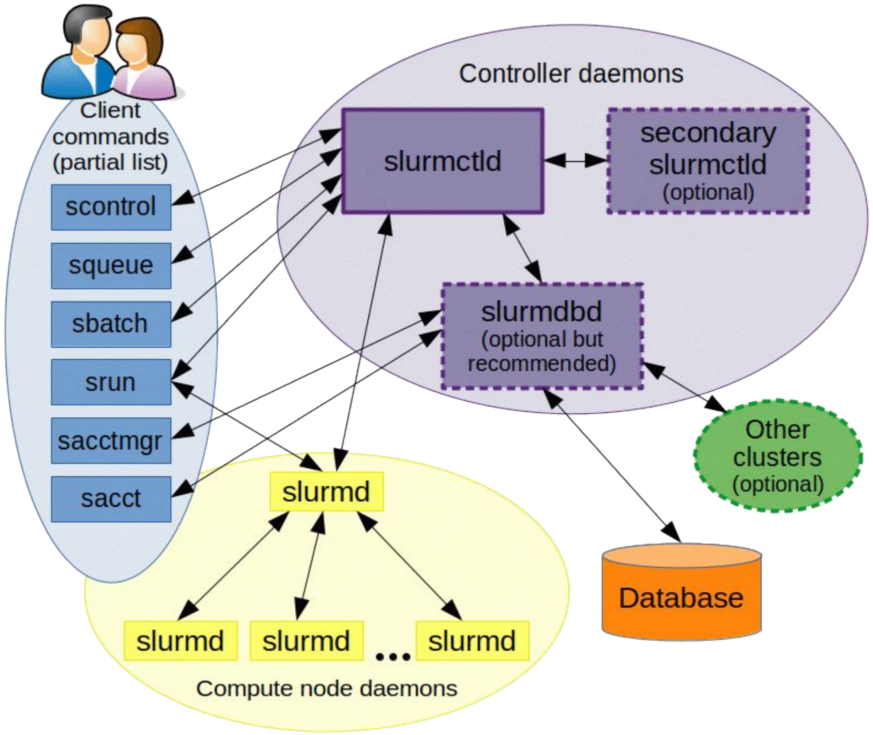
```
[someuser@bison ]$  module spider orca/5.0.2
```

```
[someuser@bison ]$  module spider orca
-----------------------------------------------------------------------------------------------
  orca:
-----------------------------------------------------------------------------------------------
        Versions:
        orca/4.2.1
        orca/5.0.1
        orca/5.0.2
-----------------------------------------------------------------------------------------------
  For detailed information about a specific "orca" package (including how to load the modules) use the
module's full name. Note that names that have a trailing (E) are extensions provided by other modules.
  For example:
        $ module spider orca/5.0.2
-----------------------------------------------------------------------------------------------
```

```
[someuser@bison ]$  module load gcc/4.8  ompi/4.1.1 orca/5.0.2
     Loading module: gcc/4.8
    Loading module: ORCA/5.0.2
```

Running jobs on HPC clusters




Client commands (partial list)
scontrol
squeue
sbatch
srun
sacctmgr
sacct
Controller daemons
slurmctld
secondary slurmctld (optional)
slurmdbd (optional but recommended)
Other clusters (optional)
Database
slurmd
slurmd
slurmd
...
slurmd
Compute node daemons

Running jobs on a cluster

- ★ Job requirements: CPUs, Memory, Time, … etc.
- ★ SLURM template: structure of a job script
- ★ Interactive jobs via salloc
- ★ Example of SLURM script: Gaussian
- ★ SLURM directives
- ★ SLURM environment variables
- ★ Examples: Serial, OpenMP, MPI, GPU
- ★ Bundle multiple jobs: job arrays and GLOST
- ★ Monitor and control your jobs: seff, scancel, sacct, …
- ★ Estimating resources: CPUs, MEM, TIME
- ★ How to pick a partition on Grex?

University of Manitoba

Scheduler: SLURM

SLURM: Simple Linux Utility for Resource Management

→ free and open-source job scheduler for Linux and Unix-like kernels

→ used by many of the world's supercomputers and computer clusters.

https://slurm.schedmd.com/overview.html

sacct - sacctmgr - salloc - sattach - sbatch - sbcast - scancel - scontrol - sdiag - seff - sh5util - sinfo - smail - smap - sprio - squeue - sreport - srun - sshare - sstat - strigger - sview

slurm
workload manager

University of Manitoba

Running jobs on a cluster

- When you connect you get interactive session on a login node:
  - Resources there are limited: used for basic operations
    - editing files, compiling codes, download or transfer data, submit and monitor jobs, run short tests {no memory intensive tests}
  - Performance can suffer greatly from over-subscription
- For interactive work, submit interactive jobs: salloc [+options]
  - SLURM uses salloc for interactive jobs [compute nodes]
  - The jobs will run on dedicated compute nodes [CPUs, GPUs]
- Submitting batch jobs for production work is mandatory: sbatch
  - Wrap commands and resource requests in a “job script”: myscript.sh
  - SLURM uses sbatch; submit a job using: sbatch myscript.sh

    sbatch [+options] myscript.sh

University of Manitoba

Job requirements

What do you need to know before submitting a job?

- Is the program available? If not, install it or ask support for help.
- What type of program are you going to run?
  - Serial, Threaded [OpenMP], MPI based, GPU, …
- Prepare your input files: locally or transfer from your computer.
- Test your program:
  - Interactive job via salloc: access to a compute node
  - On login node if the test is not memory nor CPU intensive.
- Prepare a script “myscript.sh” with the all requirements:
  - **Memory**, Number of cores, Nodes, Wall time, modules, **partition**, **accounting group**, command line to run the code.
- Submit the job and monitor it: sbatch, squeue, sacct, seff … etc

Accounting groups

- Submit Interactive job:

```
[cedar5 scratch]$  salloc --ntasks=1 --mem=4000M
salloc: error: ----------------------------------------
salloc: error: You are associated with multiple _cpu allocations...
salloc: error: Please specify one of the following accounts to submit this job:
salloc: error:   RAS default accounts: def-prof1, def-prof2
salloc: error:            RAC accounts:
salloc: error: Compute-Burst accounts:
salloc: error:          Other accounts: cc-debug,
salloc: error: Use the parameter --account=desired_account when submitting your job
salloc: error: ----------------------------------------
salloc: error: Job submit/allocate failed: Unspecified error
```

- Accounting groups: sshare -U --user <username>
  - if one accounting group, SLURM will take it by default.
  - If more than one, it should be specified via: --account={your accounting group}

University of Manitoba

Interactive jobs via salloc

```
[someuser@bison ]$  salloc --cpus-per-task=4 --mem-per-cpu=1000M --time=1:00:00
salloc: using account: def-someprof
salloc: No partition specified? It is recommended to set one! Will guess
salloc: Pending job allocation 5081294
salloc: job 5081294 queued and waiting for resources
salloc: job 5081294 has been allocated resources
salloc: Granted job allocation 5081294
salloc: Waiting for resource configuration
salloc: Nodes n063 are ready for job
        Load modules + run tests
[someuser@n063 ]$  exit
exit
salloc: Relinquishing job allocation 5081294
```

Equivalent SLURM script:

```
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1000M
#SBATCH --time=1:00:00
#SBATCH --account=def-someprof
```

University of Manitoba

Interactive jobs via salloc

```
[someuser@bison ]$  salloc --ntasks=1 --cpus-per-task=4 --mem-per-cpu=1000M
--account=def-someprof --partition=skylake --x11

salloc: using account: def-someprof
salloc: partition selected:skylake
salloc: Granted job allocation 5081297
salloc: Waiting for resource configuration
salloc: Nodes n376 are ready for job
      Load modules + run tests
[someuser@n376 ]$  exit
exit
salloc: Relinquishing job allocation 5081297
```

```
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1000M
#SBATCH --mem=4000M
#SBATCH --time=3:00:00
#SBATCH --account=def-someprof
#SBATCH --partition=skylake
```

University of Manitoba

SLURM: simple template

```
#!/bin/bash

#SBATCH --account=def-somegroup

{Add the resources and some options}


echo "Current working directory is `pwd`"
echo "Starting run at: `date`"

{Load appropriate modules if needed.}
{Command line to run your program.}


echo "Program finished with exit code $? at: `date`"
```

Script: test-job.sh

Parameters to adjust for each type of job to submit: serial, MPI, GPU

Default parameters:

- ➔ CPUs: 1
- ➔ Time: 0-3:00
- ➔ Memory: 256mb

SLURM script: Gaussian

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=7-00:00:00
#SBATCH --partition=compute

# Load appropriate modules:
module load gaussian
echo "Starting run at: `date`”
g16 < my-input.com > my-output.out
echo "Program finished with exit code $? at: `date`"
```

SLURM directives:

- Default: 1 core, 256mb, 3 hours
- **account**, number of tasks, memory per core, wall time, **partition**, …
- Other: Email notification, … etc.

Submit and monitor the job:

- `sbatch [some options] myscript.sh`
- `squeue -u $USER; sq`

Partition:

- `partition-list; sinfo --format="%20P"`
- `sinfo -s; sinfo -p <partition name>`

SLURM: some directives

#SBATCH --account=def-someprof	Use the accounting group def-someprof for jobs.
#SBATCH --ntasks=8	Request 8 tasks for MPI job; 1 for serial or OpenMP
#SBATCH --cpus-per-task=4	Number of threads (OpenMP); Threaded application
#SBATCH --ntasks-per-node=4	Request 4 tasks per-node for MPI job
#SBATCH --nodes=2	Request 2 nodes
#SBATCH --mem=4000M	Memory of 4000M for the job
#SBATCH --mem-per-cpu=2000M	Memory of 2000M per CPU
#SBATCH --partition=compute	Partition name: compute, skylake, largemem, gpu, test
#SBATCH --time=3-00:00:00	Wall time in the format: DD-HH:MM:SS

University of Manitoba

SLURM: environment variables

SLURM_JOB_NAME	User specified job name
SLURM_JOB_ID	Unique slurm job id
SLURM_NNODES	Number of nodes allocated to the job
SLURM_NTASKS	Number of tasks allocated to the job
SLURM_ARRAY_TASK_ID	Array index for this job
SLURM_ARRAY_TASK_MAX	Total number of array indexes for this job: --array-0-99
SLURM_CPUS_PER_TASK	Number of threads {OpenMP: OMP_NUM_THREADS}
SLURM_JOB_NODELIST	List of nodes on which resources are allocated to a Job
SLURM_JOB_ACCOUNT	Account under which this job is run.
SLURM_JOB_PARTITION	List of Partition(s) that the job is in.

University of Manitoba

SLURM script: serial jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=1-00:00:00
#SBATCH --partition=compute

# Load appropriate modules:
module load <software>/<version>
echo "Starting run at: `date`"
program.x [+options and arguments if any]
echo "Program finished with exit code $? at: `date`"
```

SLURM directives:

- Default: 1 core, 256mb, 3 hours
- **account**, tasks = 1, memory per core, wall time, **partition**, …
- Other: E-mail-notification, … etc.

Submit and monitor the job:

- sbatch myscript.sh
- squeue -u $USER; sq; sacct -j JOB_ID

More information:

- partition-list; sinfo --format="%20P"
- Sinfo -s; sinfo -p compute,skylake
- squeue -p compute,skylake -t R {PD}

University of Manitoba

SLURM script: OpenMP jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2000M
#SBATCH --time=1-00:00:00
#SBATCH --partition=skylake
# Load appropriate modules:
module load <software>/<version>
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
echo "Starting run at: `date`"
program.x [+options and arguments if any]
echo "Program finished with exit code $? at: `date`"
```

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2000M
#SBATCH --time=1-00:00:00
#SBATCH --partition=skylake
```

```
#SBATCH --cpus-per-task=N
#SBATCH --mem=<MEM>
```

Partitions:

- compute: N up to 12
- skylake: N up to 52
- largemem: N up to 40

University of Manitoba

SLURM script: MPI jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=96
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=1200M
#SBATCH --time=2-00:00:00
#SBATCH --partition=skylake

# Load appropriate modules:
module load intel/2019.5  ompi/3.1.4 lammps/29Sep21
echo "Starting run at: `date`”
srun lmp_grex < in.lammps
echo "Program finished with exit code $? at: `date`"
```

```
#SBATCH --nodes=8
#SBATCH --ntasks-per-node=12
#SBATCH --mem=0
#SBATCH --partition=compute
```

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=52
#SBATCH --mem=0
#SBATCH --partition=skylake
```

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=40
#SBATCH --mem=0
#SBATCH --partition=largemem
```

University of Manitoba

SLURM script: OpenMP+MPI jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=6
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=1200M
#SBATCH --time=3-00:00:00
#SBATCH --partition=compute

# Load appropriate modules:
module load <software>/<version>
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
echo "Starting run at: `date`"
srun program.x [+options and arguments if any]
echo "Program finished with exit code $? at: `date`"
```

```
#SBATCH --nodes=6
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=1200M
#SBATCH --partition=compute
```

The total memory and CPUs per node should not exceed the available resources on the nodes.

```
#SBATCH --nodes=5
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1000M
#SBATCH --partition=skylake
```

SLURM script: OpenMP+MPI jobs

```
#SBATCH --nodes=8
#SBATCH --ntasks-per-node=12
#SBATCH --cpus-per-task=1
#SBATCH --mem=0
#SBATCH --partition=compute
```

```
Job ID: 1234567
Cluster: grex
User/Group: someuser/someuser
State: COMPLETED (exit code 0)
Nodes: 8
Cores per node: 12
CPU Utilized: 156-11:07:22
CPU Efficiency: 99.22% of 157-16:44:48 core-walltime
Job Wall-clock time: 1-15:25:28
Memory Utilized: 218.00 GB (estimated maximum)
Memory Efficiency: 59.37% of 367.19 GB (45.90 GB/node)
```

The job used:

- **96 CPUs**
- **about 2400 M per core**

The job may wait longer on the queue to start: it requires 8 nodes to be available

=> Optimize the resources

```
#SBATCH --ntasks=96
#SBATCH --mem-per-cpu=2400M
#SBATCH --partition=compute
```

```
#SBATCH --ntasks=162
#SBATCH --mem-per-cpu=1200M
#SBATCH --partition=skylake
```

Bundle jobs with job arrays

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=3-00:00:00
#SBATCH --array=0-999%10
#SBATCH --partition=compute
# Load appropriate modules:
module load <software>/<version>
echo "Starting run at: `date`”
./my_code test${SLURM_ARRAY_TASK_ID}
echo "Program finished with exit code $? at: `date`"
```

- You have regularly named, independent datasets (test0, test1, test2, test3, …, test999) to process with a single software code
- Instead of making and submitting 1000 job scripts, a single script can be used with the **--array=1-999** option to **sbatch**
- Within the job script, $SLURM_ARRAY_TASK_ID can be used to pick an array element to process

  ./my_code test${SLURM_ARRAY_TASK_ID}
- When submitted, once, the script will create 1000 jobs with the index added to JobID (12345_1, … , 12345_999)
- You can use usual SLURM commands (scancel, scontrol, squeue) on either entire array or on its individual elements

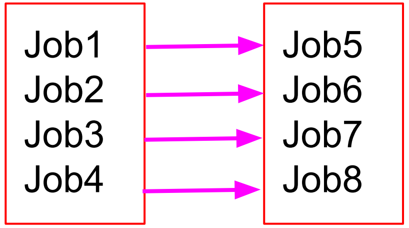
Bundle jobs with GLOST

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=4
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=1-00:00:00
#SBATCH --partition=compute

# Load appropriate modules + glost:
module load intel/15.0.5.223  ompi glost

echo "Starting run at: `date`”
srun glost_launch list_glost_tasks.txt
echo "Program finished with exit code $? at: `date`"
```

- You have many short independent jobs (job1, job2, job3, …) to process with a single software code.
- Instead of submitting and running many jobs, a single script can be used to run these jobs as MPI job.

- List of tasks: list_glost_tasks.txt

```
job1
job2
job3
job4
job5
–
job199
job200
```



Total time divided by number of commands in the list

University of Manitoba

SLURM script: GPU jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --gpu=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=6
#SBATCH --mem-per-cpu=4000M
#SBATCH --time=0-3:00:00
#SBATCH --partition=gpu
# Load appropriate modules:
module load <software>/<version>
echo "Starting run at: `date`"
program.x [+options and arguments if any]
echo "Program finished with exit code $? at: `date`"
```

SLURM directives:

- Default: 1 core, 256mb, 3 hours
- **account**, number of tasks, memory per core, wall time, **partition**, …
- Other: E-mail-notification, … etc.

Submit and monitor the job:

- sbatch [some options] myscript.sh
- squeue -u $USER

Partition:

- partition-list; sinfo --format="%20P"
- sinfo -p <partition name>

University of Manitoba

Monitor and control your jobs

```
squeue -u $USER [-t RUNNING] [-t PENDING]                      # list all current jobs.
squeue -p PartitionName [compute, skylake, largemem]    # list all jobs in a partition.
sinfo                                                 # view information about Slurm partitions.
sacct -j jobID --format=JobID,MaxRSS,Elapsed    # resources used by completed job.
sacct -u $USER --format=JobID,JobName,AveCPU,MaxRSS,MaxVMSize,Elapsed
seff  -d jobID                            # produce a detailed usage/efficiency report for the job.
sprio [-j jobID1,jobID2] [-u $USER]                             # list job priority information.
sshare -U --user $USER                                             # show usage info for user.
sinfo --state=idle; -s; -p <partition>                # show idle nodes; more about partitions.
scancel [-t PENDING] [-u $USER] [jobID]                                     # kill/cancel jobs.
scontrol show job -dd jobID                            #show more information about the job.
```

University of Manitoba

Information about the cluster

- **sinfo:** check the nodes (idle, drain, down), …

```
sinfo --state=idle        {shows idle nodes on the cluster}
sinfo --R                 {shows down, drained and draining nodes and their reason}
sinfo --Node --long       {shows more detailed information}
sinfo --p largemem        {shows more detailed information}
```

- **scontrol:** to see reservations and more

```
[~@gra-login1: ~]$ scontrol show res <Outage> --oneliner
ReservationName=Outage StartTime=2022-10-25T08:50:00 EndTime=2022-10-26T10:00:00
Duration=1-01:10:00 Nodes=gra[1-1257,1262-1325,1337-1338,1342] NodeCnt=1324
CoreCnt=44396 Features=(null) PartitionName=(null)
Flags=MAINT,IGNORE_JOBS,SPEC_NODES,ALL_NODES TRES=cpu=44396 Users=root
Groups=(null) Accounts=(null) Licenses=(null) State=INACTIVE BurstBuffer=(null) Watts=n/a
MaxStartDelay=(null)
```

University of Manitoba

SLURM: messages

★ None: the job is running (ST=R)
★ PartitionDown: one or more partitions are down (the scheduler is paused)
★ Resources: the resources are not available for this job at this time
★ Nodes required for job are DOWN, DRAINED or RESERVED for jobs in higher priority partitions: similar to Resources.
★ Priority: the job did not start because of the low priority
★ Dependency: the job did not start because it depends on another job that is not done yet.
★ JobArrayTaskLimit: the user exceeded the maximum size of array jobs
  ○
  ```
  [~@tatanka ~]$  scontrol show config | grep MaxArraySize
  MaxArraySize        = 2000
  ```
★ ReqNodeNotAvail, UnavailableNodes: n314: node not available

Estimating resources: CPUs

- ★ How to estimate the CPU resources?
  - No direct answer: it depends on the code
  - Serial code: 1 core [--ntasks=1 --mem=2500M]
  - Threaded and OpenMP: no more than available cores on a node [--cpus-per-task=12]
  - MPI jobs: can run across the nodes [--nodes=2 --ntasks-per-node=12 --mem=0].
- ★ Are threaded jobs very efficient?
  - Depends on how the code is written
  - Does not scale very well
  - Run a benchmark and compare the performance and efficiency.
- ★ Are MPI jobs very efficient?
  - Scale very well with the problem size
  - Limited number of cores for small size: when using domain decomposition
  - Run a benchmark and compare the efficiency.

University of Manitoba

Estimating resources: memory

- ★ How to estimate the memory for my job?
  - No direct answer: it depends on the code
  - Java applications require more memory in general
  - Hard to estimate the memory when running R, Python, Perl, …
- ★ To estimate the memory, run tests:
  - Interactive job, ssh to the node and run top -u $USER {-H}
  - Start smaller and increase the memory
  - Use whole memory of the node; seff <JOBID>; then adjust for similar jobs
  - MPI jobs can aggregate more memory when increasing the number of cores
- ★ What are the best practices for evaluation the memory:
  - Run tests and see how much memory is used for your jobs {seff; sacct}
  - **Do not oversubscribe the memory** since it will affect the usage and the waiting time: accounting group charged for resources reserved and not used properly.

University of Manitoba

Estimating resources: run time

- How to estimate the run time for my job?
  - No direct answer: it depends on the job and the problem size
  - See if the code can use checkpoints
  - For linear problems: use a small set; then estimate the run time accordingly if you use more steps (extrapolate).
- To estimate the time, run tests:
  - Over-estimate the time for the first tests and adjust for similar jobs and problem size.
- What are the best practices for time used to run jobs?
  - Have a good estimation of the run time after multiple tests.
  - Analyse the time used for previous successful jobs.
  - Add a margin of 15 to 20 % of that time to be sure that the jobs will finish.
  - Do not overestimate the wall time since it will affect the start time: longer jobs have access to smaller partition on the cluster (Compute Canada clusters).

University of Manitoba

Optimizing jobs: mem and CPU

★ How to estimate the run time for my job?
  - No direct answer: it depends on the job and the problem size
  - See if the code can use checkpoints
  - For linear problems: use a small set; then estimate the run time accordingly if you use more steps (extrapolate).

★ To estimate the time, run tests:
  - Over-estimate the time for the first tests and adjust for similar jobs and problem size.

★ What are the best practices for time used to run jobs?
  - Have a good estimation of the run time after multiple tests.
  - Analyse the time used for previous successful jobs.
  - Add a margin of 15 to 20 % of that time to be sure that the jobs will finish.
  - Do not overestimate the wall time since it will affect the start time: longer jobs have access to smaller partition on the cluster (the Alliance clusters).

Memory and CPU efficiencies: seff

Output from seff command for a job {OpenMP} that asked for 24 CPUs and 187 GB of memory on cedar:

```
Job ID: 123456789
Cluster: cedar
User/Group: someuser/someuser
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 24
CPU Utilized: 38-14:26:22
CPU Efficiency: 38.46% of 100-08:45:36 core-walltime
Job Wall-clock time: 4-04:21:54
Memory Utilized: 26.86 GB
Memory Efficiency: 14.37% of 187.00 GB
```

Successful job

Low CPU efficiency: 40 %
Better performance with 8 CPU

Used less memory: 15 %

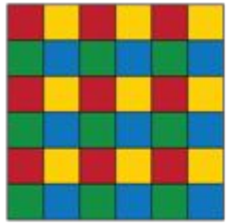
billing=46,cpu=24,mem=187G,node=1

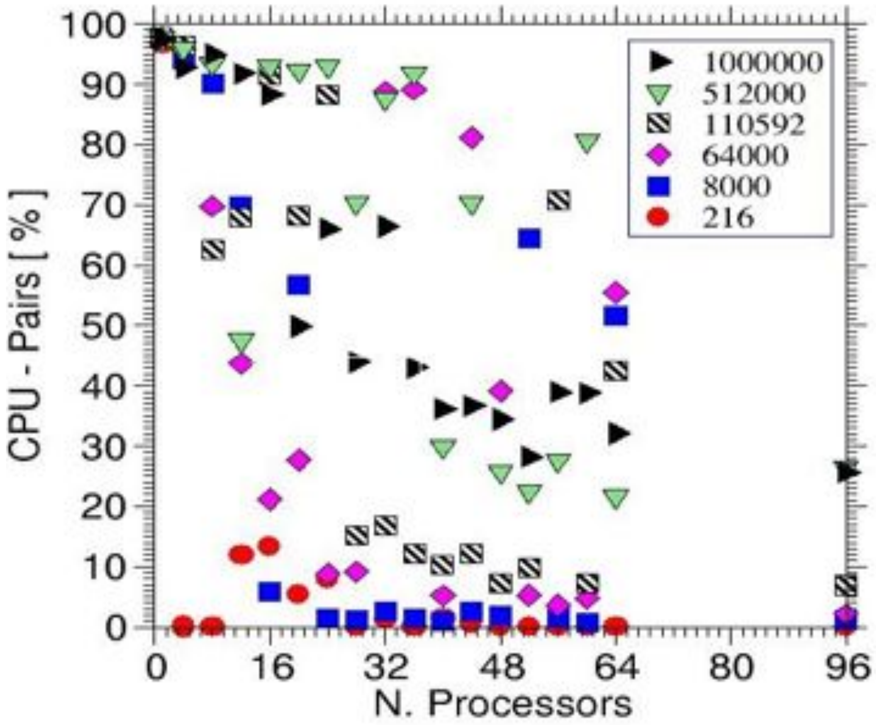
Optimization:
Better performance with 8 CPU
Memory: 4000 M per core [32 GB]

```
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --mem-per-cpu=4000M
```

Performance tests: MD

Domain decomposition





- ★ Size, shape of the system.
- ★ Number of processors.
- ★ size of the small units.
- ★ correlation between the communications and the number of small units.
- ★ Reduce the number of cells to reduce communications.

How to get most of the scheduler?

The key is to know what resources are available on a given HPC machine, and to adjust your requests accordingly.

- ★ It is up to the users to figure it out: documentation; tests, …
- ★ Know what partitions are there, and what are their limits: sinfo, …
- ★ Know what is the hardware (how many CPUs per node, how much memory per CPU available, …. documentation for each cluster
- ★ Know if your code is efficient for a given set of resources: benchmarks
- ★ Know time limits and estimate runtime of your jobs
  - ○ comes after some trials and errors, with experience
- ★ Make sure your application obeys the SLURM resource limits

How to pick a CPU partition on Grex?

Many jobs are submitted to skylake partition and asing for large memory: by over-subscribing the memory, many CPUs will stay idle [low usage of].

Partition	Nodes	Cores	Total	Memory	MEM/CPU
compute	312	12	3456	46 GB	3.8 GB
largemem	12	40	480	376 GB	9.4 GB
skylake	42	52	2184	96 GB	1.6 GB

Some tips for usage optimization:

- Run tests and check the memory usage {seff}
- Adjust the memory for similar jobs
- Submit with appropriate resources {no more}.

Output from: partition-list

```
PARTITION     CPUS(A/I/O/T)
compute*      2280/300/1280/3860
largemem      480/0/0/480
skylake       781/1455/0/2236
```

Partitions and memory:

compute: many nodes {312} and many CPUs {3456}
serial and MPI jobs with memory per CPU around 4 GB.

skylake: only 42 nodes but many CPUs {2184}
serial and MPI jobs with memory per CPU around 1.6 GB.

largemem: few nodes {12}, 480 CPUs
serial and MPI jobs with memory per CPU around 9 GB.

Skylake partition shows 781 allocated CPUs and 1455 idle CPUs. These CPUs are idle and can not run other job because all the memory was allocated to other jobs.

University of Manitoba

Thank you for your attention

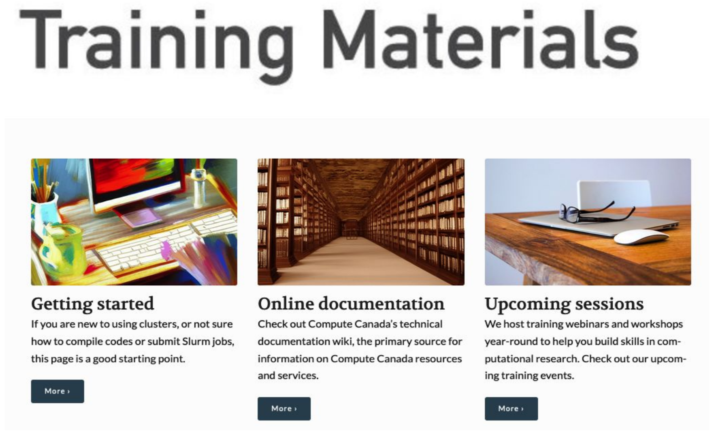
Any question?

More readings

- The Alliance [Compute Canada]: https://docs.alliancecan.ca/wiki/Main_Page
- CCDB: https://ccdb.computecanada.ca/security/login
- CC Software: https://docs.alliancecan.ca/wiki/Available_software
- Running Jobs: https://docs.alliancecan.ca/wiki/Running_jobs
- SLURM: https://slurm.schedmd.com/
- PuTTy: http://www.putty.org/
- MobaXterm: https://mobaxterm.mobatek.net/
- X2Go: https://wiki.x2go.org/doku.php
- Grex: https://um-grex.github.io/grex-docs/

→ WG training material: https://training.westdri.ca/

→ Help and support {Grex+Alliance}: support@tech.alliancecan.ca



Training Materials

Getting started
If you are new to using clusters, or not sure how to compile codes or submit Slurm jobs, this page is a good starting point.
More ›

Online documentation
Check out Compute Canada's technical documentation wiki, the primary source for information on Compute Canada resources and services.
More ›

Upcoming sessions
We host training webinars and workshops year-round to help you build skills in computational research. Check out our upcoming training events.
More ›

University of Manitoba

Demonstration: MD simulation

- Serial job
- MPI job:
  - 4; 8; 16; 32; 48; 64; 96
- Job array: parameter sweep
  - data in one directory
  - data in multiple directories
- Estimation of wall wall time
- Memory efficiency
- CPU efficiency

```
module load intel/2019.5  ompi/3.1.4 lammps/29Sep21
srun lmp_grex < in.melt > log_lammps_output.txt
```

in.melt

```
# 3d Lennard-Jones melt
units           lj
atom_style      atomic
lattice         fcc 0.8442
region          box block 0 50 0 50 0 50
create_box      1 box
create_atoms    1 box
mass            1 1.0
velocity        all create 3.0 87287
pair_style      lj/cut 2.5
pair_coeff      1 1 1.0 1.0 2.5
neighbor        0.3 bin
neigh_modify    every 20 delay 0 check no
fix             1 all nve
thermo          250
run             10000
write_data      config.end_melt
```

University of Manitoba

Serial job

```
[~@bison ]$  seff 5080534
Job ID: 5080534
Cluster: grex
User/Group: someuser/someuser
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 01:28:33
CPU Efficiency: 99.87% of 01:28:40
core-walltime
Job Wall-clock time: 01:28:40
Memory Utilized: 274.48 MB
Memory Efficiency: 3.43% of 7.81 GB
```

- ➔ Job completed
- ➔ CPU Efficiency: 99.87%
- ➔ Wall time: 01:28:40
- ➔ Memory Utilized: 274.48 MB

Steps: 10000 (iterations)
Wall time: 1:30 to 2:00
Memory: 300 mb to 500 mb

Steps: 10000 x 10
Wall time: {1:30 to 2:00} x 10
Memory: 300 mb to 500 mb

Time breakdown

```
Loop time of 5316.35 on 1 procs for 10000 steps with 500000 atoms

Performance: 812.587 tau/day, 1.881 timesteps/s
99.8% CPU use with 1 MPI tasks x no OpenMP threads

MPI task timing breakdown:
Section |  min time  |  avg time  |  max time  |%varavg| %total
---------------------------------------------------------------
Pair    | 4617.5     | 4617.5     | 4617.5     |   0.0 | 86.86
Neigh   | 517.75     | 517.75     | 517.75     |   0.0 |  9.74
Comm    | 35.556     | 35.556     | 35.556     |   0.0 |  0.67
Output  | 0.11397    | 0.11397    | 0.11397    |   0.0 |  0.00
Modify  | 119.55     | 119.55     | 119.55     |   0.0 |  2.25
Other   |            | 25.83      |            |       |  0.49
```

MPI job

CPUs	CPU Efficiency	CPU time	Run time [s]	Performance tau/day	Pair Interactions	Communication time [%]
1	99.87%	5317	5317	812.587	86.86	-
4	99.76%	6200	1550	2787	84.50	3.13
8	99.09%	6312	789	5479	78.13	10.77
16	99.21%	7360	460	9388	67.74	21.35
32	98.32%	5984	187	23136	76.97	10.32
48	97.56%	5904	123	35220	74.85	12.63
64	95.17%	5888	92	47175	73.62	14.52
96	95.03%	5760	60	71874	73.24	15.16

Job array

```
#!/bin/bash
#SBATCH --account=def-kerrache
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=4000M
#SBATCH --time=0-8:00
#SBATCH --partition=compute
#SBATCH --array=0-9%2

echo "Starting run at: `date`"
module load intel/2019.5  ompi/3.1.4 lammps/29Sep21

lmp_grex < in.melt-${SLURM_ARRAY_TASK_ID}.txt >
log_lammps_array-${SLURM_ARRAY_TASK_ID}.txt

echo "Program finished with exit code $? at: `date`"
```

- Files: n.melt-0.txt, …. In.melt-9.txt
- Job array with 10 elements
- Run a maximum of 2 at a time
- All the data in one directory:
- use appropriate names to avoid data overlapping

Job array: another example

- Files: n.melt-0.txt, …. In.melt-9.txt; array with 10 elements; Run a maximum of 2 at a time
- All the data in one directory: use appropriate names to avoid data overlapping

```
lmp_grex < in.melt-${SLURM_ARRAY_TASK_ID}.txt > log_lammps_array-${SLURM_ARRAY_TASK_ID}.txt
```

- Directories: 0, …. 9; each directory has a an input file: in.melt
- Job array with 10 elements
- Run a maximum of 2 at a time
- Output in different directories: the data may have the same name.

```
cd ${SLURM_ARRAY_TASK_ID}
lmp_grex < in.melt > log_lammps_array-${SLURM_ARRAY_TASK_ID}.txt
```