University of Manitoba

Start Guide for Using Grex efficiently

Access to Grex, Using Software and Running jobs on Grex

UofM-Autumn-Session 2024

Oct 31st-Nov 01st, 2024

Ali Kerrache

HPC Analyst

University of Manitoba

Outline

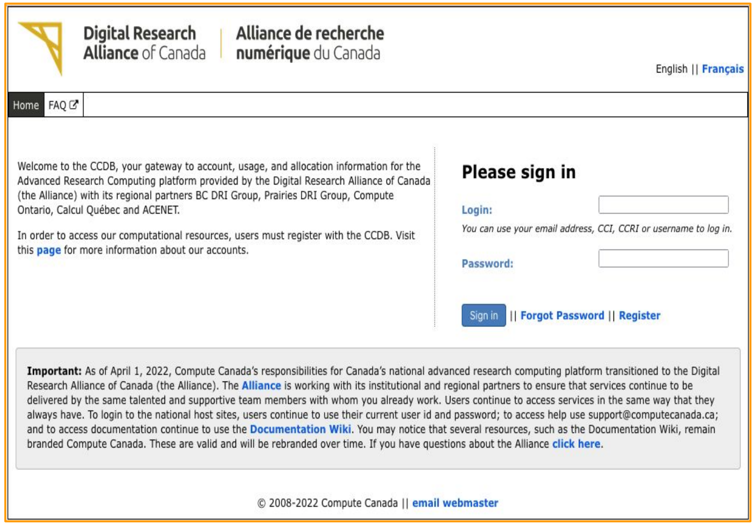
- ➔ Access to Grex
- ➔ HPC software
- ➔ Running jobs on Grex

Access to Grex/Alliance

Step 1:

Principal Investigator (PI) or sponsor

Faculty member registers in the Alliance Database (CCDB): https://ccdb.alliancecan.ca/security/login


Digital Research Alliance of Canada | Alliance de recherche numérique du Canada
English || Français
Home FAQ

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 2: sponsored users:

Master's student, Doctoral student, PostDoctoral fellow, Researcher, External collaborators, … etc.

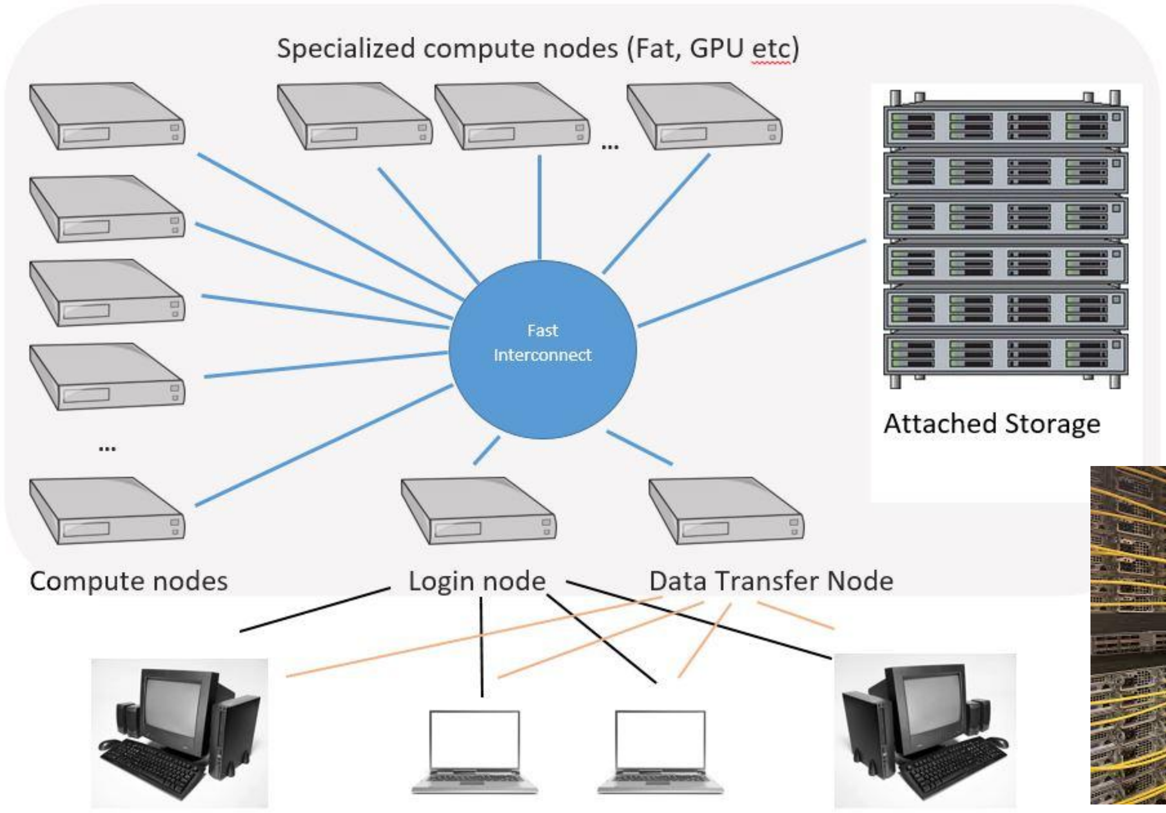
Once PI's account is approved, sponsored users can register as group members (CCRI: abc-123-01).

- ➔ One account per user and only the role can change over time.
- ➔ All accounts are renewed once a year (Spring)

Secure your account:

- **Do not share credentials**
- Use SSH keys
- Add MFA {mandatory}

What is an HPC cluster?



Resources on Grex: CPUs

Partition	Nodes [CPUs]	Cores	Total	Memory	Max Wall Time
skylake	43	52	2236	187 GB	21 days
chrim	4	192	768	750 GB	21 days
chrimlm	1	192	192	1500 GB	21 days
genlm	3	192	192	1500 GB	21 days
genoa	27	192	5184	750 GB	21 days
largemem	12	40	480	380 GB	21 days
mcordcpu	5	168	840	1500 GB	21 days
–	95	-	9892	-	-

https://um-grex.github.io/grex-docs/running-jobs/slurm-partitions/

Resources on Grex: GPUs

Partition	Nodes [GPUs]	Cores	Total	Memory	Wall Time
gpu	2 [4 V100 - 32 GB]	32	64	187 GB	7 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
mcordgpu; -b	4 [NVIDIA A30 - 24 GB]	32	128	500 GB	1 days / 7 days
test	-	18	18	500 GB	23 hours
–	–	–	402	–	–

Contributed partitions: stamps, livi, agro, mcordgpu ➔ Group owning the hardware.

Backfill partitions: stamps-b, livi-b, agro-b, mcordgpu-b ➔ Other users.

University of Manitoba

Contributed & backfill partitions

Contributed partitions:

- **GPU:** stamps, livi, agro, mcordgpu
- **CPU:** mcordcpu, chrim, chrimlm

Backfill partitions:

- **GPU:** stamps-b, livi-b, agro-b, mcordgpu-b
- **CPU:** genoacpu-b

How contributed and backfill partitions work?

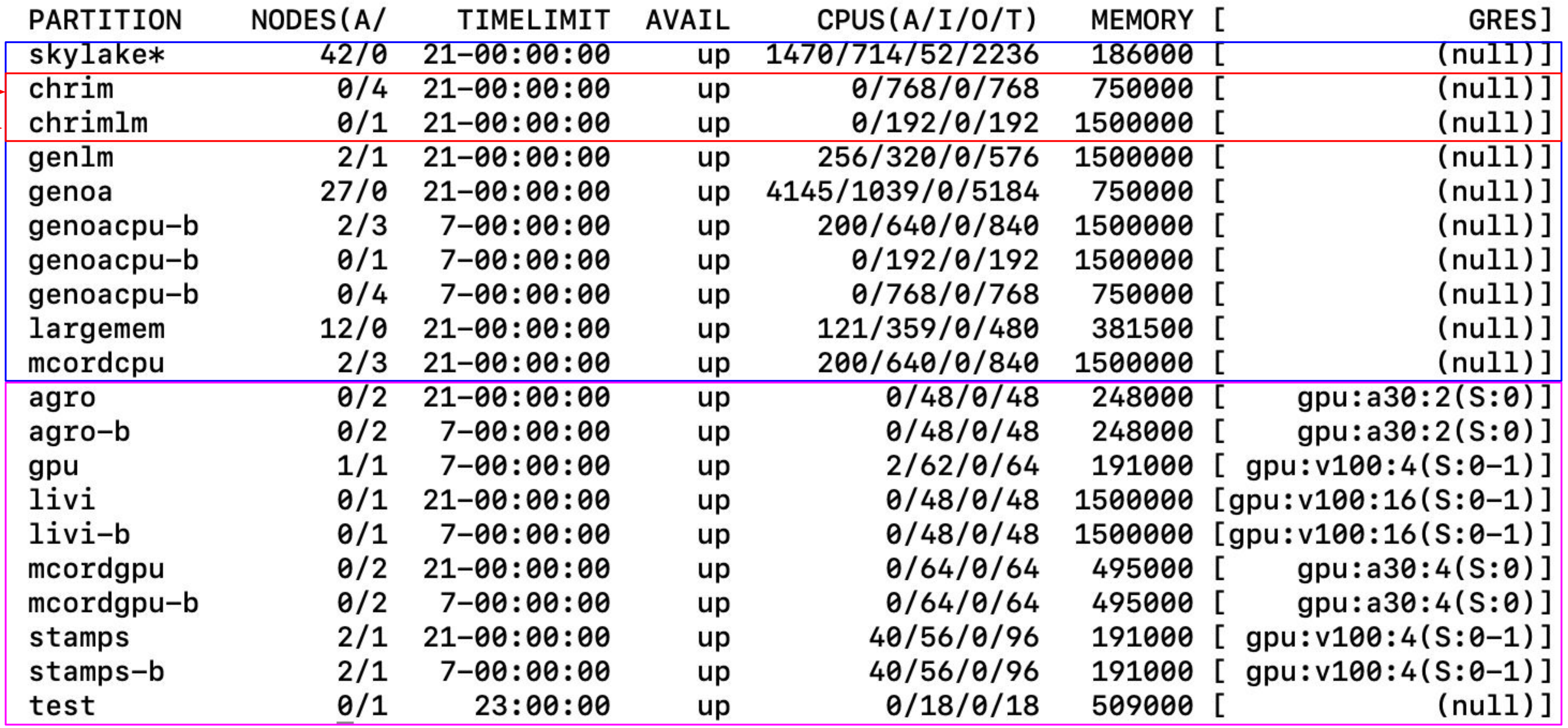
- Hardware owned by particular groups.
- The group owner have a preferential access to their partitions: chrim, chrimlm
- If not used by the owner, the partitions can be used by other users: genoacpu-b

What if the contributed partition is busy running jobs from other users?

- Even if genoacpu-b is busy to run other jobs, the group owner still has priority.
- The jobs using genoacpu-b will be preempted to free the resources for the group to use their own hardware {chrim; chrimlm}

https://um-grex.github.io/grex-docs/running-jobs/contributed-systems/

Custom script: partition-list



```
PARTITION      NODES(A/    TIMELIMIT  AVAIL       CPUS(A/I/O/T)    MEMORY [             GRES]
skylake*           42/0  21-00:00:00     up  1470/714/52/2236    186000 [           (null)]
chrim               0/4  21-00:00:00     up       0/768/0/768    750000 [           (null)]
chrimlm             0/1  21-00:00:00     up       0/192/0/192   1500000 [           (null)]
genlm               2/1  21-00:00:00     up     256/320/0/576   1500000 [           (null)]
genoa              27/0  21-00:00:00     up  4145/1039/0/5184    750000 [           (null)]
genoacpu-b          2/3   7-00:00:00     up     200/640/0/840   1500000 [           (null)]
genoacpu-b          0/1   7-00:00:00     up       0/192/0/192   1500000 [           (null)]
genoacpu-b          0/4   7-00:00:00     up       0/768/0/768    750000 [           (null)]
largemem           12/0  21-00:00:00     up     121/359/0/480    381500 [           (null)]
mcordcpu            2/3  21-00:00:00     up     200/640/0/840   1500000 [           (null)]
agro                0/2  21-00:00:00     up         0/48/0/48    248000 [     gpu:a30:2(S:0)]
agro-b              0/2   7-00:00:00     up         0/48/0/48    248000 [     gpu:a30:2(S:0)]
gpu                 1/1   7-00:00:00     up         2/62/0/64    191000 [ gpu:v100:4(S:0-1)]
livi                0/1  21-00:00:00     up         0/48/0/48   1500000 [gpu:v100:16(S:0-1)]
livi-b              0/1   7-00:00:00     up         0/48/0/48   1500000 [gpu:v100:16(S:0-1)]
mcordgpu            0/2  21-00:00:00     up         0/64/0/64    495000 [     gpu:a30:4(S:0)]
mcordgpu-b          0/2   7-00:00:00     up         0/64/0/64    495000 [     gpu:a30:4(S:0)]
stamps              2/1  21-00:00:00     up        40/56/0/96    191000 [ gpu:v100:4(S:0-1)]
stamps-b            2/1   7-00:00:00     up        40/56/0/96    191000 [ gpu:v100:4(S:0-1)]
test                0/1     23:00:00     up         0/18/0/18    509000 [           (null)]
```

C
P
U

G
P
U

University of Manitoba

How to use your own partitions?

To use your own partitions, add:
--partition=chrim or chrimlm to your salloc or sbatch commands or to your scripts.

From command line:

```
salloc --partition=chrim {+options}
salloc --partition=chrimlm {+options}
sbatch --partition=chrim job-script.sh
sbatch --partition=chrim job-script.sh
```

Inside a job script:

```
#SBATCH --partition=chrim
```
or
```
#SBATCH --partition=chrimlm
```

```
[~@yak ~]$  sinfo -p chrim
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
chrim        up 21-00:00:0      4   idle n[424-427]

[~@yak ~]$  sinfo -p chrimlm
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
chrimlm      up 21-00:00:0      1   idle n423

[~@yak ~]$ scontrol show partition chrim
[~@yak ~]$ squeue -t R -p chrim
[~@yak ~]$ squeue -t R -p chrimlm
[~@yak ~]$ squeue -t PD -p chrim
```

University of Manitoba

Workflow on HPC clusters

Connect to a cluster

Linux/Mac:

⇒ ssh client

⇒ OOD

Windows:

⇒ Putty

⇒ … etc

Transfer files

Linux, Mac:

⇒ scp, sftp, rsync

Windows:

⇒ WinScp

⇒ FileZilla

⇒ … etc

HPC work

★ Connect

★ Transfer files

★ Compile codes

★ Test jobs

★ Run jobs

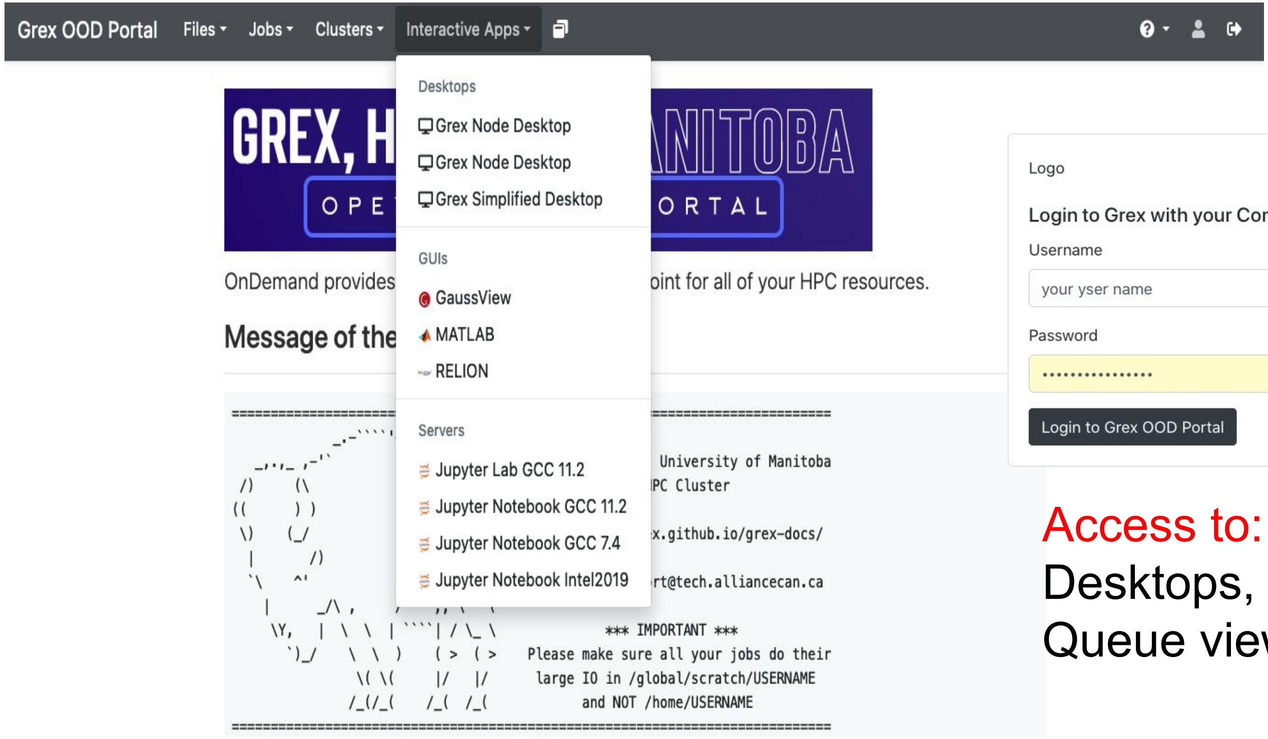
★ Analyze data

★ Visualisation

OpenOnDemand: remote web access to supercomputers

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

OOD: a web portal for Grex



Grex OOD Portal | Files ▾ | Jobs ▾ | Clusters ▾ | Interactive Apps ▾

Desktops
- Grex Node Desktop
- Grex Node Desktop
- Grex Simplified Desktop

GUIs
- GaussView
- MATLAB
- RELION

Servers
- Jupyter Lab GCC 11.2
- Jupyter Notebook GCC 11.2
- Jupyter Notebook GCC 7.4
- Jupyter Notebook Intel2019

GREX, H… …ANITOBA
OPE… …ORTAL

OnDemand provides … oint for all of your HPC resources.

Message of the…

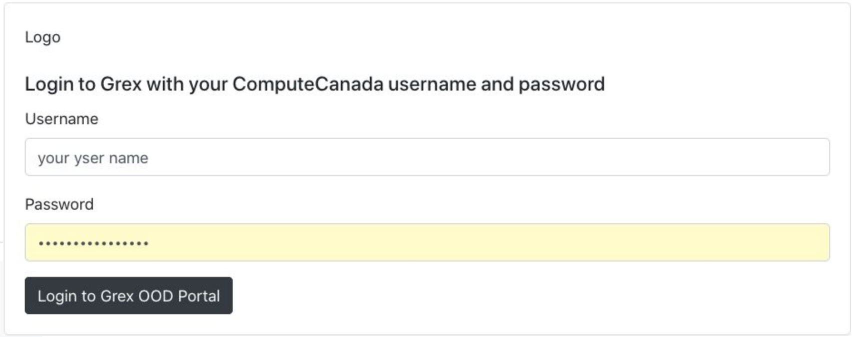
```
======================                         =======================
                                               University of Manitoba
                                              PC Cluster

                                              x.github.io/grex-docs/

                                              rt@tech.alliancecan.ca

                                          *** IMPORTANT ***
                                  Please make sure all your jobs do their
                                   large IO in /global/scratch/USERNAME
                                        and NOT /home/USERNAME
=====================================================================
```

https://zebu.hpc.umanitoba.ca



Logo

Login to Grex with your ComputeCanada username and password

Username: your yser name

Password: •••••••••••••••••

Login to Grex OOD Portal

Access to:
Desktops, Jupyter, Terminal, Jobs, Queue view, Running jobs, …

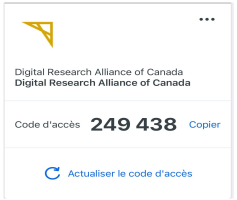
Hostname: zebu.hpc.umanitoba.ca

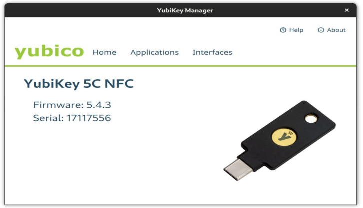
https://um-grex.github.io/grex-docs/ood/

Improve security: SSH keys, MFA

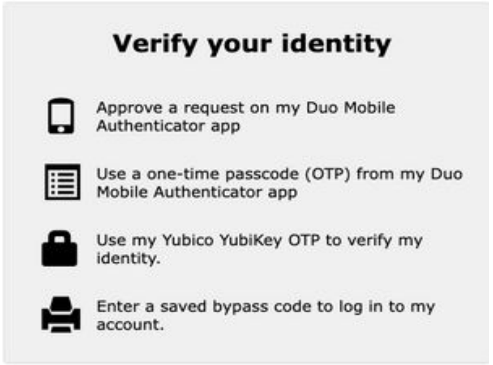
- ★ **Multifactor authentication:**
  - Mandatory for all staff
  - Mandatory for all users
- ★ **Grex**
  - ssh keys in CCDB
  - VPN for OpenOnDemand
  - MFA for Grex

```
[name@server ~]$ ssh cluster.computecanada.ca
Duo two-factor login for name
Enter a passcode or select one of the following options:
1. Duo Push to My phone (iOS)
Passcode or option (1-1):abcdefghijklmnopqrstuvwxyz
Success. Logging you in...
```

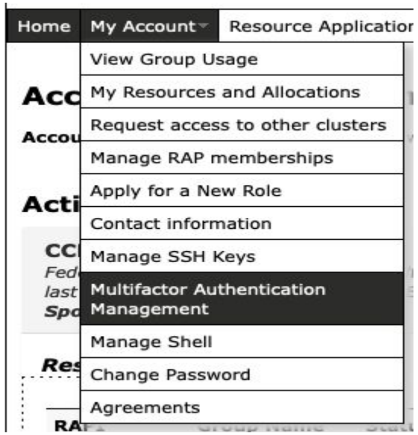

Digital Research Alliance of Canada
Digital Research Alliance of Canada
Code d'accès 249 438 Copier
Actualiser le code d'accès



YubiKey 5C NFC
Firmware: 5.4.3
Serial: 17117556


Verify your identity

- Approve a request on my Duo Mobile Authenticator app
- Use a one-time passcode (OTP) from my Duo Mobile Authenticator app
- Use my Yubico YubiKey OTP to verify my identity.
- Enter a saved bypass code to log in to my account.



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

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/40T            250k/2M
```

Over quota

Space under home directory

Inode under project def-somep

```
[someuser@yak ~]$  diskusage_report
                    Description (FS)     Space (U/Q)     # of files (U/Q)
                 /home (someuser)       226M/104G         2381/500k
 ~~/global/scratch (someuser)          519G/4294G          27k/1000k~~
         /project (def-someprof)       3201G/5242G         17k/2000k
```

- - home

~~- - scratch~~

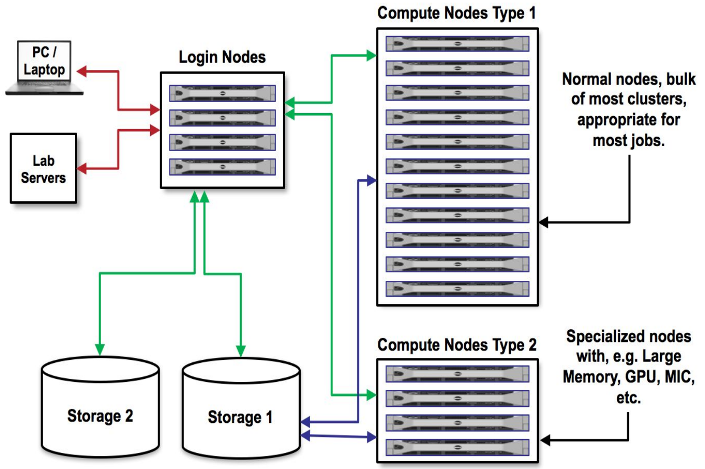
- - project

HPC Software

HPC Software

- ★ Software distribution on Grex
- ★ Why modules? How to find modules?
- ★ Software stacks on Grex
- ★ Build software from sources
  - ○ R packages
  - ○ Python packages
  - ○ Perl modules
  - ○ configure/make
  - ○ cmake/make
- ★ Singularity/Apptainer (separate talk)

HPC Cluster:
- ★ Hardware
- ★ Network
- ★ Software


PC / Laptop
Lab Servers
Login Nodes
Compute Nodes Type 1
Normal nodes, bulk of most clusters, appropriate for most jobs.
Compute Nodes Type 2
Specialized nodes with, e.g. Large Memory, GPU, MIC, etc.
Storage 2
Storage 1

University of Manitoba

Why modules?

★ Why modules?

https://um-grex.github.io/grex-docs/software/using-modules/

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

```
[someuser@yak ]$  module list
–
Currently Loaded Modules:
  1) SBEnv (S)

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

★ 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

Software stacks on Grex

★ Grex environment [default]: SBEnv
- ○ no module loaded by default.
- ○ use module spider <name of the software> to search for modules
- ○ Compilers: {GCC, Intel, …}, MKL, PETSc, … etc.
- ○ Gaussian, ANSYS, MATLAB, … etc.

★ The Alliance (Compute Canada) environment [optional]: CCEnv
- ○ Switch to CCEnv; load a standard environment; choose the architecture[avx2, avx512], use module spider <soft>

```
~$ module load CCEnv
~$ module load arch/avx512
~$ module load StdEnv/2023
~$ module load gcc/12.3.0 samtools
```

Using local software stack: SBEnv

```
~@yak: module load arch/avx512
~@yak: module load gcc/13.2.0 samtools/1.20
```

Modules on Grex

- Compilers/Libraries and more:
  - Compilers: GCC [8.5 - 13.2]; Intel [2019, 2023], … etc.
  - Libraries: HDF5, PETSc, GSL, MKL, Libxc, Boost, ...
  - Gaussian, ANSYS, MATLAB, VASP, ORCA, MCR, Java, Python, R, … etc.
  - LAMMPS, GROMACS, QE, OpenBABEL, … etc.
- Software maintenance on Grex and Alliance clusters:
  - We install programs and update modules on request from users.
  - Search for a program using "module spider <name of your program>"
  - If not installed, ask for support "support@tech.alliancecan.ca"
  - We will install the module and/or update the version.
  - For commercial software, contact us before you purchase the code:
    - to check license type.
    - see if it will run under Linux environment, … etc.

University of Manitoba

Building software

- Local installation [user's directory: home, project]:
  - R packages; Julia packages, Perl modules
  - Python packages: virtual environment
  - Home made programs and commercial software.
- Installation with:
  - make; make test {check}; make install
  - configure; make; make test {check}; make install
  - cmake; make; make test {check}; make install
- Java applications: jar files
- Singularity and/or Aptainer: {separate talk}
  - build the image and run your program from the container

Local installation

- ★ R packages: minimal installation
  - ○ R as modules: users can install the packages in their home directory.
- ★ Python as modules: python and scipy-stack
  - ○ users can install the packages needed in their home directory.
- ★ Perl as module:
  - ○ users can install the packages needed in their home directory.
- ★ Other software installed locally:
  - ○ Home made programs {up to a user or a group}
  - ○ Restricted and licensed software that can not be distributed
  - ○ Custom software: patch from a user, changing parts of the code, … etc.

University of Manitoba

Local installation: R packages

R packages: rgdal, adegenet, stats, rjags, dplyr, … etc.

Choose a module version: module spider r

Load R and dependencies (gdal, geos, jags, gsl, udunits… etc):

```
module load gcc r gdal udunits
```

Launch R and install the packages:

```
~$ R
> install.packages(”sp")
'lib =/cvmfs/soft.computecanada.ca/easybuild/{..}/R/library"' is not writable
Would you like to use a personal library instead? (yes/No/cancel) yes
Would you like to create a personal library ‘~/R/{…}’ to install packages into? (yes/No/cancel) yes
--- Please select a CRAN mirror for use in this session ---
> install.packages(”dplyr")
```

University of Manitoba

Local installation: Python

- ★ Load the modules:
  - ○ module load python
- ★ Create a virtual environment
  - ○ virtualenv ~/my_venv
- ★ Activate the virtual environment
  - ○ source ~/my_venv/bin/activate
- ★ Update pip
  - ○ pip install --no-index --upgrade pip
- ★ Install the packages
  - ○ pip install pandas
  - ○ pip install -r requirements.txt
  - ~~○ python setup.py install~~

```
module load gcc python/3.1
virtualenv ~/my_venv
source ~/my_venv/bin/activate
pip install cutadapt
deactivate
```

```
module load gcc python/3.11.2
source ~/my_venv/bin/activate
cutadapt [+options]
deactivate
```

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

University of Manitoba

Local installation: Perl

Example: Hash::Merge; Logger::Simple; MCE::Mutex; threads …

Load Perl module: module load perl

Install the the first package using cpan or cpanm:

```
~$ cpan install YAML
Would you like to configure as much as possible automatically? [yes] yes
What approach do you want?  (Choose 'local::lib', 'sudo' or 'manual')
[local::lib] local::lib
Would you like me to append that to /home/$USER/.bashrc now? [yes] yes
```

Install the rest of the packages using cpan or cpanm:

```
~$ cpan install Hash::Merge
~$ cpan install Logger::Simple
~$ cpan install MCE::Mutex
```

Installation with make: STAR

- ★ Download the code {wget; curl; git clone; …}:

```
wget https://github.com/alexdobin/STAR/archive/refs/tags/2.7.10b.tar.gz
```

- ★ Unpack the code: `tar -xvf 2.7.10b.tar.gz`
- ★ Load GCC compiler: `module load gcc`
- ★ Compile the code:

```
cd  STAR-2.7.10b/source
make
```

- ★ Copy the binaries and set the path:

```
mkdir -p ~/software/star/2.7.10b/bin
cp STAR  ~/software/star/2.7.10b/bin
export PATH=$PATH:${HOME}/software/star/2.7.10b/bin
```

Installation with configure/make

- ★ Download and unpack the code: wget, … gunzip, … etc.
- ★ Load the modules and dependencies: module load gcc ompi fftw
- ★ Configure the program
  - ○ If configure not included, run: autoreconf -fvi [to generate it].
  - ○ ./configure --help [to see the different options].
  - ○ ./configure --prefix=<path to install dir> {+other options}
- ★ Compile and test:
  - ○ make; make -j4
  - ○ make check; make test
- ★ Install the program:
  - ○ make install

Installing: treemix

- ★ Download the source files:
  wget https://bitbucket.org/nygcresearch/treemix/downloads/treemix-1.13.tar.gz
- ★ Unpack the source files: tar -xvf treemix-1.13.tar.gz
- ★ Change the directory: cd treemix-1.13/
- ★ Load the modules: module load gcc boost
- ★ Configure: ./configure --prefix=/home/$USER/software/treemix/1.13
- ★ Compile and install: make && make test && make install
- ★ Set a path: export PATH=$PATH:$HOME/software/treemix/1.13/bin
- ★ Usage in a job script:

```
module load gcc boost
export PATH=$PATH:$HOME/software/treemix/1.13/bin
treemix {+options if any}
```

Example with cmake/make

- ★ Download and unpack the code: wget, … gunzip, … etc.
- ★ Load the modules and dependencies: module load gcc ompi fftw
- ★ Configure the program: you may need to load cmake module
  - mkdir build && cd build
  - cmake .. --help [to see the different options].
  - cmake .. -DCMAKE_INSTALL_PREFIX=installdir {+other options}
- ★ Compile and test:
  - make; make -j8
  - make check; make test
- ★ Install the program:
  - make install

Java applications

★ Download and unpack the code
★ Load java module: module load java
★ Run the code

★ Example: Trimmomatic
  ○ wget http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/Trimmomatic-0.39.zip
  ○ unzip Trimmomatic-0.39.zip

★ Run the code

```
module load java
java -jar <path to>/trimmomatic-0.39.jar {+options if any}
```

Singularity/Apptainer

- ★ Alternative for running software: difficult to build from source
- ★ Possibility to convert Docker images to singularity.
- ★ Singularity/Apptainer installed on all clusters {no Docker for security reasons}
- ★ Build the image:

```
module load singularity
singularity build qiime2-2021.11.sif docker://quay.io/qiime2/core:2021.11
```

- ★ Run the code via singularity:

```
singularity exec -B $PWD:/home -B /global/scratch/someuser:/outputs  \
-B /global/scratch/someuser/path/to/inputs:/inputs <path to qiime2-2021.11.sif>   \
qiime feature-classifier fit-classifier-naive-bayes \
--i-reference-reads  /outputs/some_output_feature.qza \
--i-reference-taxonomy /outputs/some_output_ref-taxonomy.qza \
--o-classifier /outputs/some_output_classifier.qza
```

Running jobs on Grex

University of Manitoba

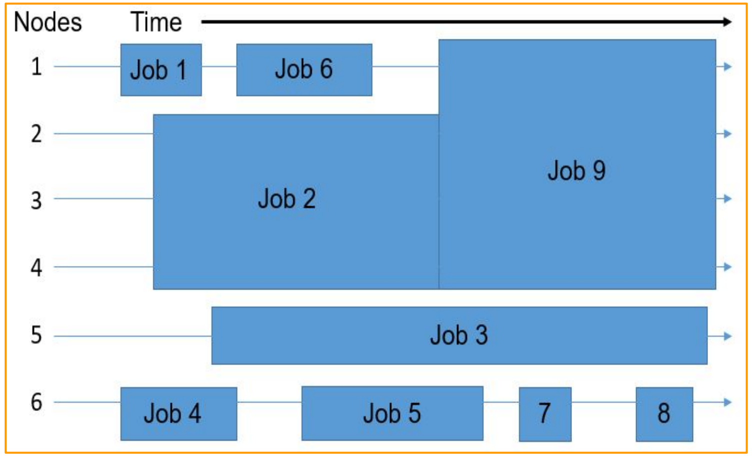
Scheduler: SLURM

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



University of Manitoba

Running jobs on Grex

★ Job requirements: CPUs, Memory, Time, … etc.

★ SLURM template: structure of a job script

★ Interactive jobs via salloc

★ Example of SLURM script.

★ 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?*

slurm workload manager

University of Manitoba

Interactive jobs

- ★ When you connect you get interactive session on a login node:
  - ○ Limited resources: to be used with care 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 [CPU, GPU]
- ★ 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 the 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 and monitor the jobs: sbatch, squeue, sacct, seff … etc

Accounting groups

```
salloc --ntasks=1 --mem=4000M --account=def-prof1
```

- ★ Submit Interactive job:

```
[~yak ]$  salloc --ntasks=1 --mem=4000M
salloc: error: ----------------------------------------
salloc: error: You have more than one account:
salloc: error:  - account # 1: use sbatch/salloc --account=def-prof1
salloc: error:  - account # 2: use sbatch/salloc --account=def-prof2
salloc: error: Please pick one 'sbatch/salloc --account=_VALUE_' option from the list above ^^^
salloc: error: Or set 'export SBATCH_ACCOUNT=_VALUE_' (or 'export
SALLOC_ACCOUNT=_VALUE_') to one of the above accounts
salloc: error: Job submit/allocate failed: Invalid account or account/partition combination specified
```

- ★ 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}

Interactive jobs via salloc

```
[someuser@yak ]$  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 n365 are ready for job
        Load modules + run tests
[someuser@n365 ]$  exit
exit
salloc: Relinquishing job allocation 5081294
```

Equivalent SLURM script:

```
#!/bin/bash
#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@yak ]$  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
```

```
#!/bin/bash
#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: basic 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

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=chrim

# Load appropriate modules:
module load <dep> <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 chrim
- squeue -p chrim -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=chrim
# 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=chrim
```

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

Partitions:

- chrim: N up to 192
- skylake: N up to 52
- largemem: N up to 40

Bundle many jobs: job array

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

University of Manitoba

Bundle many jobs: job array

```
#!/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=genoa
# 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 many jobs: GLOST

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

# Load appropriate modules + glost:
module load intel openmpi 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
```

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=N]
  - MPI jobs: can run across the nodes [--nodes=2 --ntasks-per-node=52 --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?
  - Scales very well with the problem size
  - Limited number of cores for small size: when using domain decomposition
  - Run a benchmark and compare the efficiency.

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.

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).

University of Manitoba

Memory & 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
```

University of Manitoba

How to pick a CPU partition on Grex?

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

Partition	Nodes	Cores	Total	Memory	MEM/CPU
genoa	27	192	5184	750 GB	3.9 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)
Genoa         4225/959/0/5184
largemem      480/0/0/480
skylake       781/1455/0/2236
```

Partitions and memory:

genoa: only 27 nodes but many CPUs {5184}
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 4 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.

How to get most of the scheduler?

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

- ★ It is up to the users to go through the documentation and run tests, …
- ★ Know what partitions are there, and what are their limits: sinfo, …
- ★ Know about 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.

Summary about HPC workflow

- Account and active role:
  - CCDB
- Have a look to the documentation:
  - Hardware, available tools, …
  - policies?
  - login nodes
  - storage, …
- Tools to connect and transfer files
- Access to storage: home, scratch, project
- Access to a program to use:
  - Install the program or ask for it.
  - Use the existing modules

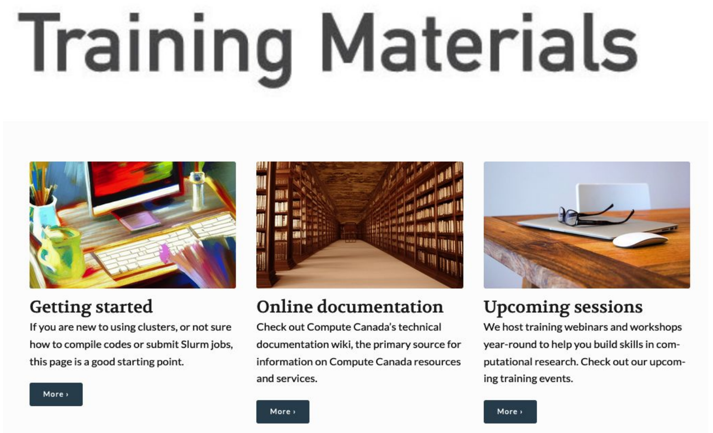
- Test jobs:
  - Login node
  - Interactive job via salloc
- Write a job script:
  - Slurm directives
  - Modules
  - Command line to run the code
- Monitor jobs:
  - sacct; seff, optimize jobs
- Analyze data:
  - Post processing
  - Visualization

More readings

- The Alliance [Compute Canada]: https://docs.alliancecan.ca/wiki/Main_Page
- CCDB: https://ccdb.alliancecan.ca/security/login
- MFA: https://docs.alliancecan.ca/wiki/Multifactor_authentication
- PuTTy: http://www.putty.org/
- 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

Thank you for your attention

Any question?

Demonstration

- Connect to Grex via ssh
- Transfer a directory using scp
- Install a software
- Build an image using singularity {covered on a separate talk}
- Submit jobs:
  - Serial job
  - Array job
- Monitor jobs: seff, squeue, … etc

University of Manitoba

Additional Slides

What to get from your account?

Access to all clusters:

- **Grex:** available only for UManitoba users and their collaborators
- **cedar**, **graham**, **beluga**, **narval**, niagara: canadian researchers.
- **Cloud**: on request.
- **Nextcloud**, Globus, … etc.

Opportunistic usage:

- CPU
- GPU
- Storage [1 TB to 10 TB]

Resource Allocations Competition:

- CPU, GPU, Storage, VCPUs, …
- Implementation on April each year.

Storage: file systems and quota

the Alliance [Compute Canada]:

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

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

Grex:

/home/$USER:

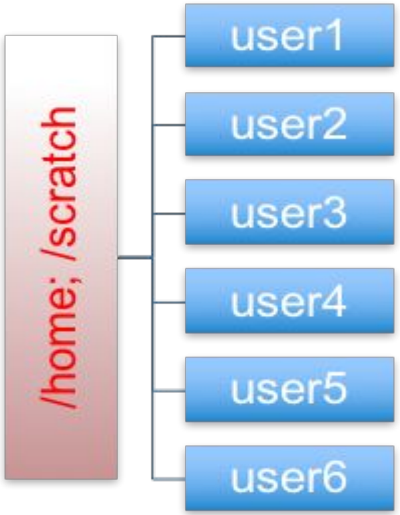
100 GB per user

~~/global/scratch/$USER:~~

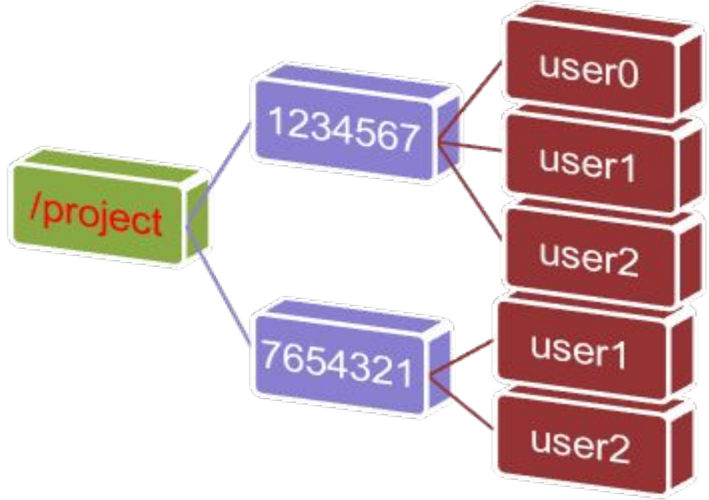
~~**4** TB, no backup, no purge.~~

/project

backup, no purge.



Project: projects/def-professor/$USER



1 TB per group; extension up to 40 TB

Backup; Allocatable via RAC (>40 TB)

University of Manitoba

Connect, transfer files, …

- ★ **ssh** **=>** Secure Shell [connect to a remote machine].
- ★ **scp** **=>** Secure Copy [copy file to/from a remote host].
- ★ **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.

- ★ **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
```

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

★ **password**

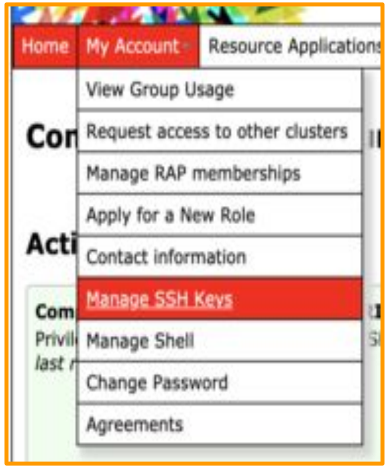
★ **ssh keys**

Improve security: SSH keys

- ★ Generate ssh keys: https://docs.alliancecan.ca/wiki/SSH_Keys#Generating_an_SSH_Key
  - Private key:
    - keep it in your computer: ~/.ssh/
    - do not share it or copy it to any cluster.
  - Public key:
    - Copy the key to remote machine
    - ssh-copy-id -i mykey someuser@niagara.computecanada.ca
- ★ Copy the public key to:
  - Remote machine [cluster]
  - **CCDB**
- ★ Mandatory to connect to niagara

```
ssh -i <path to your key> someuser@niagara.computecanada.ca
```

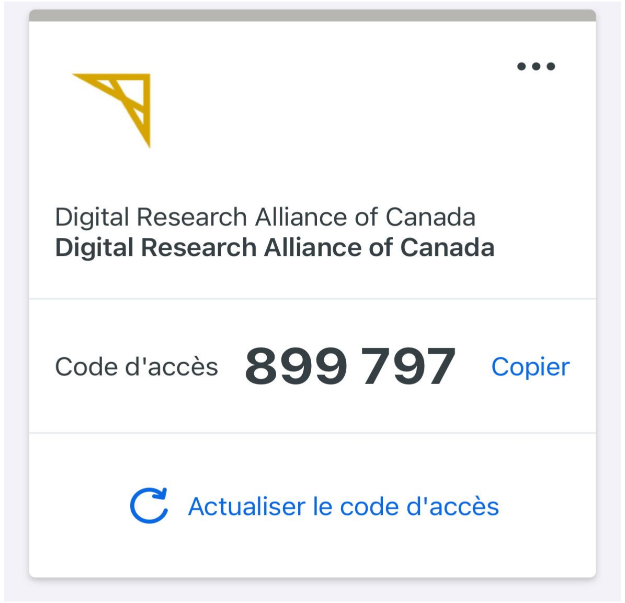
- ★ Enabled on Grex



Options for MFA

★ **Options:**

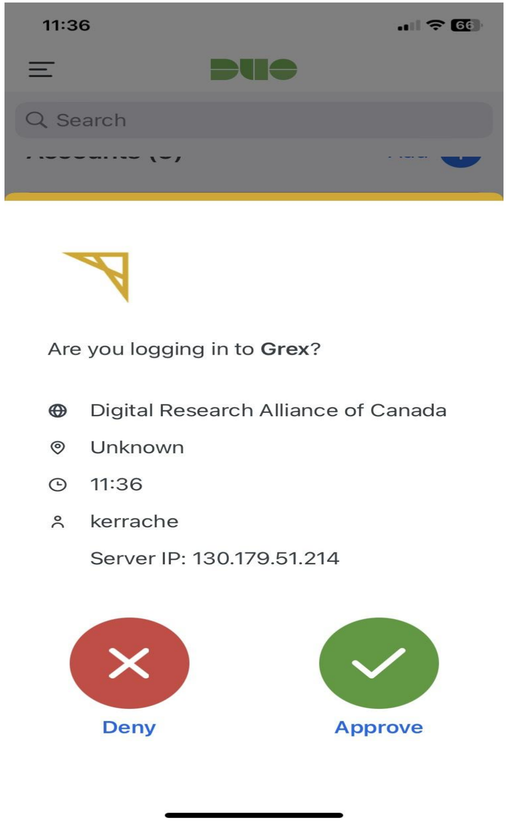
- Ubikey
- Phone
- Access code



Digital Research Alliance of Canada
Digital Research Alliance of Canada

Code d'accès 899 797 Copier

Actualiser le code d'accès



11:36

Search

Are you logging in to **Grex**?

Digital Research Alliance of Canada
Unknown
11:36
kerrache
Server IP: 130.179.51.214

Deny Approve

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

File transfer: scp, sftp, rsync, …

Terminal: Linux; Mac; CygWin; PuTTy, … etc.

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.

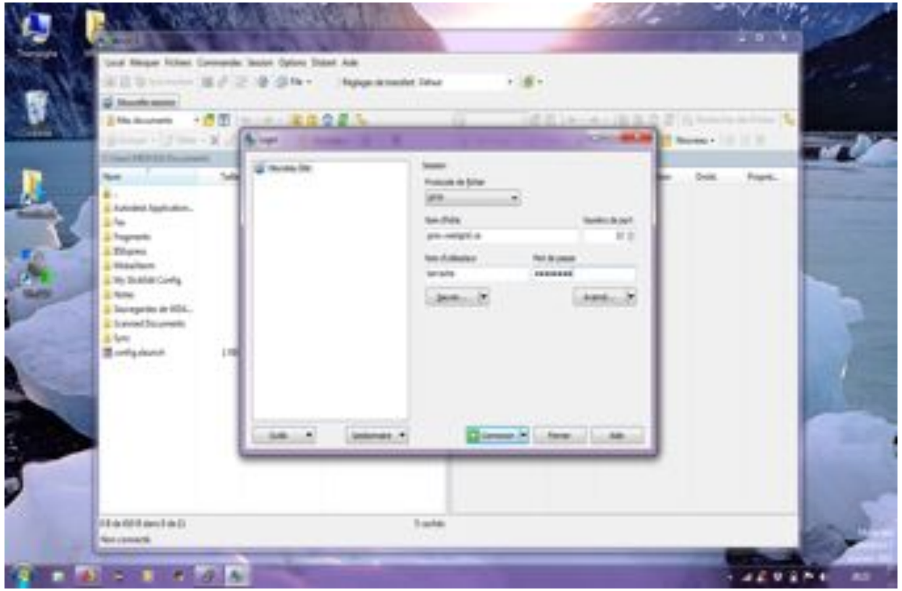
username@grex.hpc.umanitoba.ca:/home/username/{Your_Dir}; ~/{Your_Dir}

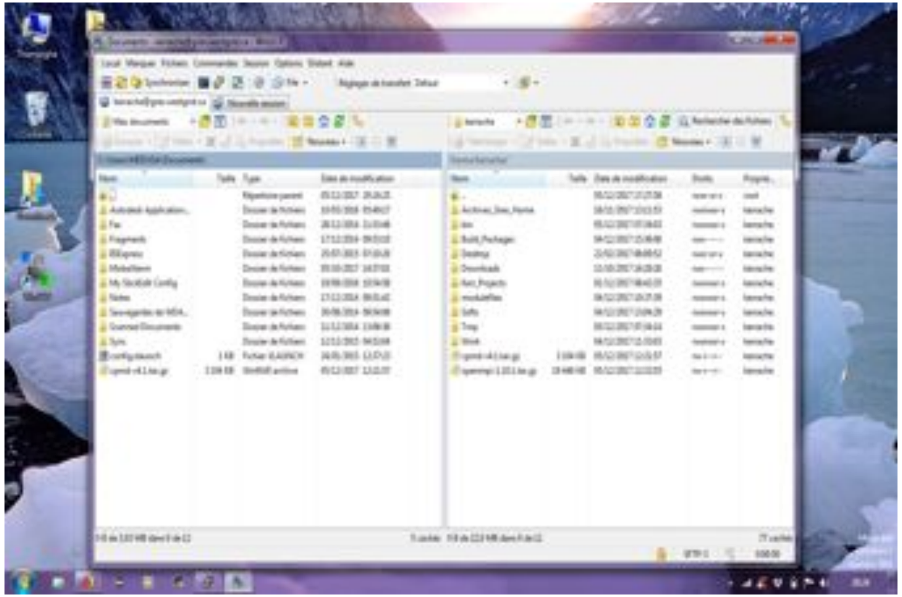
username@grex.hpc.umanitoba.ca:/global/scratch/username/{Your_Dir}

```
[~@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.





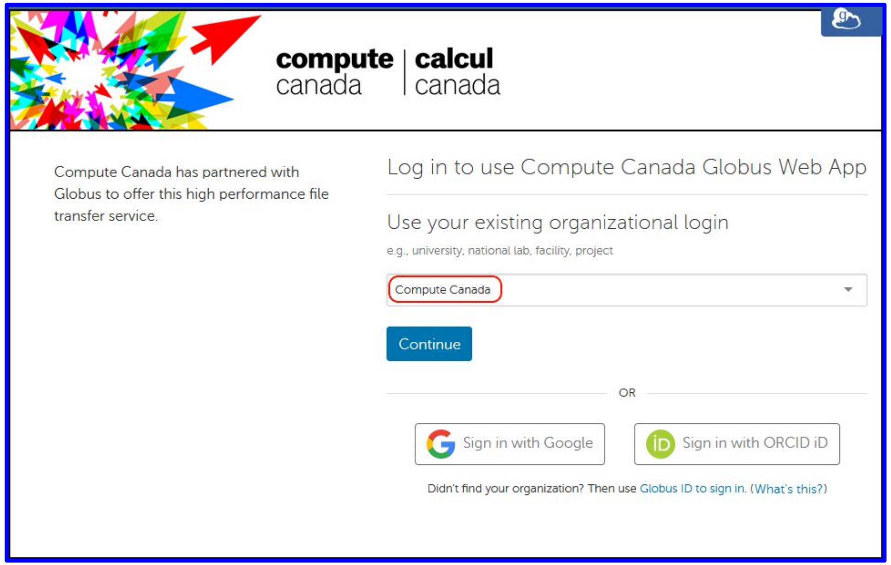
- Navigate on your local machine.
- Navigate on remote machine.
- Copy your files (works on both ways).

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

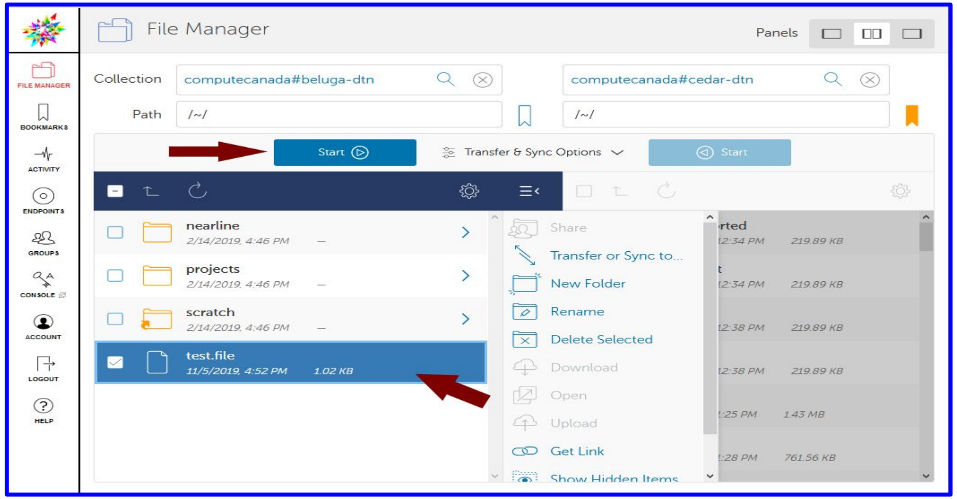
Continue

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?)



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

Path: /~/ | /~/

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

https://um-grex.github.io/grex-docs/connecting/data-transfer/

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

Software layers

User layer: Python packages, Perl and R modules, home made codes, …

Software stacks: modules for Intel, PGI, OpenMPI, CUDA, MKL, high-level applications. Multiple architectures (sse3, avx, avx2, avx512)

Nix or gentoo: GNU libc, autotools, make, bash, cat, ls, awk, grep, etc.

Gray area: Slurm, Lustre client libraries, IB/OmniPath/InfiniPath client libraries (all dependencies of OpenMPI) in Nix {or gentoo} layer, but can be overridden using PATH & LD_LIBRARY_PATH.

OS: kernel, daemons, drivers, libcuda, anything privileged (e.g. the sudo command): always local. Some legally restricted software too (VASP).

User

Analysts

Sys. Admin

More about slurm and jobs

SLURM: most used 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	–nodes=<Min>-<Max> Request 2 nodes
#SBATCH --mem=1500M	Memory of 1500M for the job
#SBATCH --mem-per-cpu=2000M	Memory of 2000M per CPU
#SBATCH --partition=compute	**GREX:** 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 script: MPI jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --nodes=2-4
#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=1
#SBATCH --ntasks-per-node=192
#SBATCH --mem=0
#SBATCH --partition=genoa
```

```
#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=2
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1200M
#SBATCH --time=3-00:00:00
#SBATCH --partition=skylake

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

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=1200M
#SBATCH --partition=genoa
```

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

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1000M
#SBATCH --partition=skylake
```

Script: by node versus by core

```
#SBATCH --nodes=5
#SBATCH --ntasks-per-node=16
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=4000M
#SBATCH --partition=skylake
```

```
Job ID: 1234567
Cluster: grex
User/Group: someuser/someuser
State: COMPLETED (exit code 0)
Nodes: 5
Cores per node: 16
CPU Efficiency: 97.48% of 65-02:16:00 core-walltime
Job Wall-clock time: 19:31:42
Memory Utilized: 151.68 GB (estimated maximum)
Memory Efficiency: 48.0% of 312.0 GB (3.95 GB/core)
```

The job used:

- **80 CPUs**
- **about 4000 M per core**

The job may wait longer on the queue to start: it requires 5 nodes to be available
=> Optimize the resources

```
#SBATCH --ntasks=80
#SBATCH --mem-per-cpu=2000M
#SBATCH --partition=skylake
```

```
#SBATCH --ntasks=160
#SBATCH --mem-per-cpu=1000M
#SBATCH --partition=skylake
```

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.
```

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 its 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

  -
    ```
    [~@yak ~]$  scontrol show config | grep MaxArraySize
    MaxArraySize        = 2000
    ```
- ReqNodeNotAvail, UnavailableNodes: n365: node not available

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=0-999%10
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	Accounting group under which this job is running.
SLURM_JOB_PARTITION	List of Partition(s) that the job is in.

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)
```

Information about a partition

```
[~@bison ~]$  sinfo -p largemem
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
largemem        up 14-00:00:0  5      mix n[328-331,333]
largemem        up 14-00:00:0  6  alloc n[326-327,334-337]
largemem        up 14-00:00:0  1   idle n332
```

```
[~@bison ~]$  scontrol show partition largemem --oneliner
PartitionName=largemem AllowGroups=ALL AllowAccounts=ALL AllowQos=normal,high
AllocNodes=aurochs,tatanka,bison,wisent,yak,n[001-316],g32[1-5],g338,g383,n[326-337],n[33
9-381] Default=NO QoS=N/A DefaultTime=03:00:00 DisableRootJobs=NO ExclusiveUser=NO
GraceTime=0 Hidden=NO MaxNodes=UNLIMITED MaxTime=14-00:00:00 MinNodes=0
LLN=NO MaxCPUsPerNode=UNLIMITED Nodes=n[326-337] PriorityJobFactor=0
PriorityTier=1 RootOnly=NO ReqResv=NO OverSubscribe=NO OverTimeLimit=NONE
PreemptMode=OFF State=UP TotalCPUs=480 TotalNodes=12 SelectTypeParameters=NONE
JobDefaults=(null) DefMemPerCPU=7000 MaxMemPerNode=UNLIMITED
TRESBillingWeights=CPU=2.0,Mem=0
```

Queued jobs: squeue

```
[someuser@yak ~]$ squeue

[someuser@yak ~]$ squeue -u $USER

[someuser@yak ~]$ sq

[someuser@yak ~]$ squeue -u <someuser>

[someuser@yak ~]$ squeue -t R

[someuser@yak ~]$ squeue -t PD

[someuser@yak ~]$ squeue -p compute,skylake -t R

[someuser@yak ~]$ squeue -j <jobid>
```

Monitor queued jobs:

- Per user
- Job ID
- Per partition
- Running jobs
- Pending job
- Combine two or more from the above.
- .. etc.

Queued jobs: scontrol

```
[someuser@yak ~]$ scontrol show job 1234567 --oneliner
JobId=1234567 JobName=run-lmp-serial.sh UserId=someuser(3333333)
GroupId=someuser(3333333) MCS_label=N/A Priority=491351 Nice=0 Account=def-someprof
QOS=normal JobState=RUNNING Reason=None Dependency=(null) Requeue=0 Restarts=0
BatchFlag=1 Reboot=0 ExitCode=0:0 RunTime=01:23:18 TimeLimit=12:00:00 TimeMin=N/A
SubmitTime=2023-11-03T09:26:35 EligibleTime=2023-11-03T09:26:35
AccrueTime=2023-11-03T09:26:35 StartTime=2023-11-03T09:26:51 EndTime=2023-11-03T21:26:51
Deadline=N/A SuspendTime=None SecsPreSuspend=0 LastSchedEval=2023-11-03T09:26:51
Scheduler=Backfill Partition=compute AllocNode:Sid=yak:174565 ReqNodeList=(null)
ExcNodeList=(null) NodeList=n204 BatchHost=n204 NumNodes=1 NumCPUs=1 NumTasks=1
CPUs/Task=1 ReqB:S:C:T=0:0:*:* TRES=cpu=1,mem=4000M,node=1 Socks/Node=*
NtasksPerN:B:S:C=0:0:*:* CoreSpec=* MinCPUsNode=1 MinMemoryCPU=4000M
MinTmpDiskNode=0 Features=(null) DelayBoot=00:00:00 OverSubscribe=OK Contiguous=0
Licenses=(null) Network=(null) Command=/home/someuser/Workshop/Serial_Job/run-lmp-serial.sh
WorkDir=/home/someuser/Serial_Job StdErr=/home/someuser/Serial_Job/slurm-1234567.out
StdIn=/dev/null StdOut=/home/someuser/Serial_Job/slurm-1234567.out Power=
```

Jobs and nodes by partition

```
[someuser@yak ~]$  squeue -p skylake
[someuser@yak ~]$  squeue -p skylake -t PD
[someuser@yak ~]$  squeue -p skylake -t R

[someuser@yak ~]$  sinfo -p skylake
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
skylake    up 21-00:00:0  1  inval n349
skylake    up 21-00:00:0  3  down* n[352,359-360]
skylake    up 21-00:00:0  1  drain n375
skylake    up 21-00:00:0  26    mix n[339-342,346-347,350-351,356-358,366-374,376-381]
skylake    up 21-00:00:0  12  alloc n[343-345,348,353-355,361-365

[someuser@yak ~]$  sinfo -p skylake --state=down
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
skylake    up 21-00:00:0  3  down* n[352,359-360]
```