University of Manitoba

Digital Research Alliance of Canada

Introduction to High Performance Computing step by step:

From getting an account to running jobs

UofM-Spring-Workshop 2022

May 4th-5th, 2022

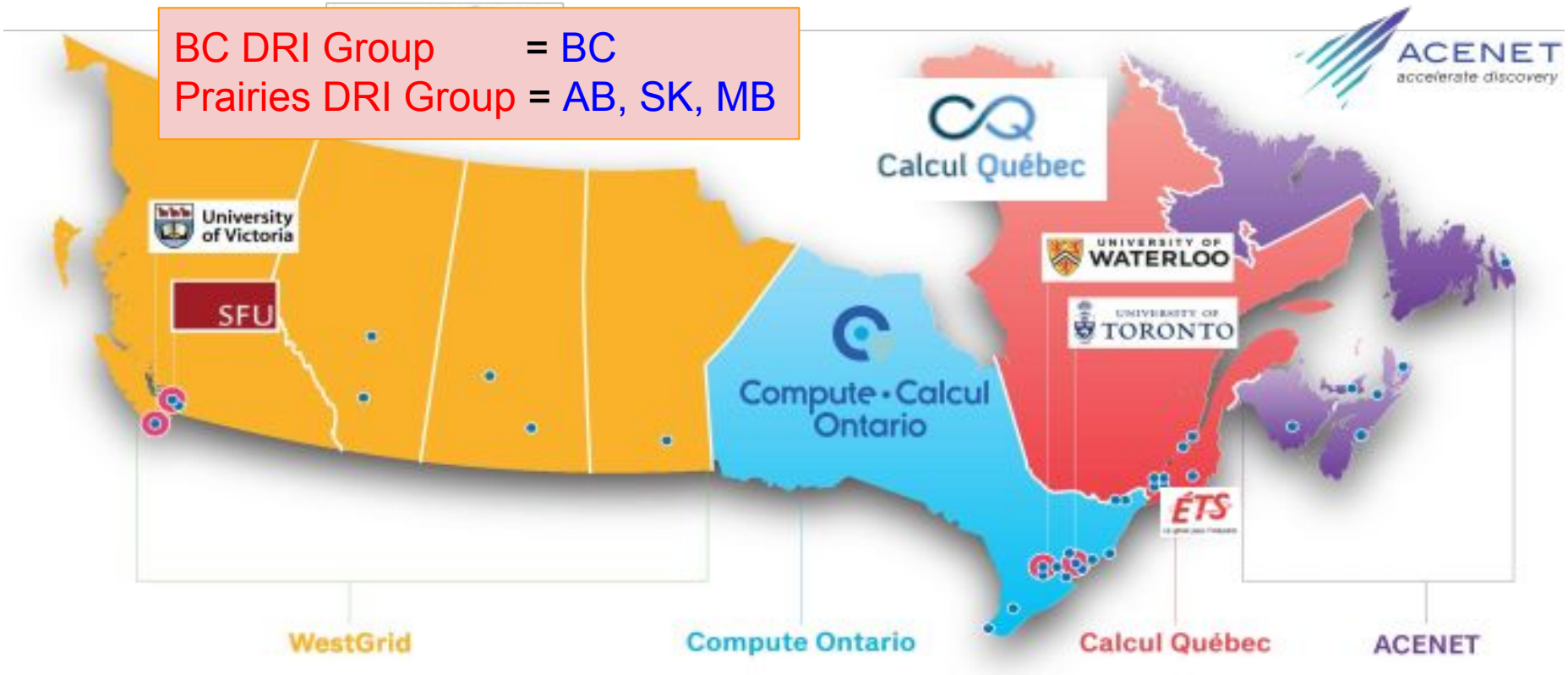
Ali Kerrache

University of Manitoba

Outline

- ★ Available resources:
  - The Alliance: cedar, graham, beluga, narval, niagara, cloud.
  - Grex [**UofM**]: available hardware
- ★ All you need to know about using HPC resources:
  - Get an account (and active role): CCDB
  - Linux shell (Terminal, command line)
  - Connect to a cluster: ssh client, PuTTY, MobaXterm, X2Go, OOD
  - Transfer files: scp, rsync, sftp, WinSCP, FileZilla…
  - Install programs and/or use existing modules
  - Submit and monitor jobs: sbatch, salloc, squeue, seff … etc.

The Alliance [Compute Canada]



University of Manitoba

The Alliance's 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.computecanada.ca/wiki/Compute_Canada_Documentation

University of Manitoba

Available nodes on Grex: partitions

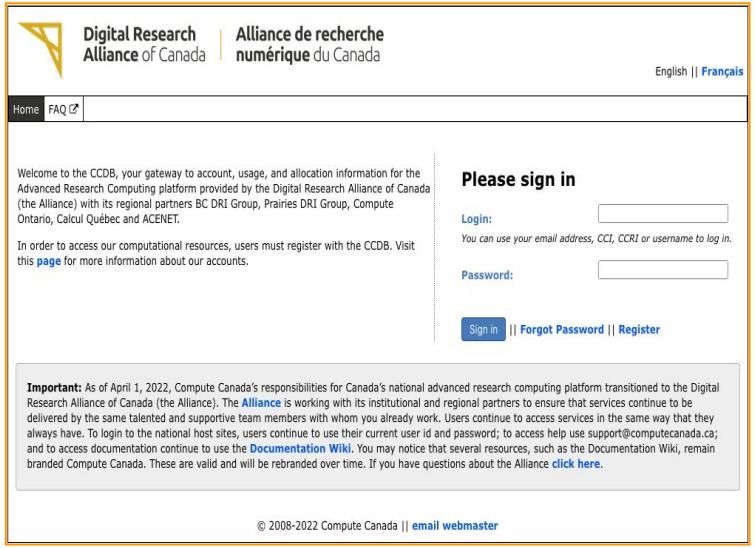
Partition	Nodes [CPUs/GPUs]	Cores	Total	Memory	Wall Time
compute[1]	312	12	3456	46 GB	21 days
largemem	12	40	480	376 GB	14 days
skylake	42	52	2184	96 GB	21 days
gpu	2 [4 V100 - 32 GB]	32	64	187 GB	3 days
stamps; -b	3 [4 V100 - 16 GB]	32	96	187 GB	21 days / 7 days
livi; -b	[16 V100 - 32 GB]	48	48	1.5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

Structure of HPC clusters: Grex

Access to Alliance clusters / Grex



The Alliance: Rapid Access Service

10 TB of storage / cluster.

RAC for storage > 10 TB.

Send an email to: support@computecanada.ca

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).

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.

University of Manitoba

HPC: workflow and tools

Connect to a cluster

Linux:

ssh client, X2Go

Mac:

ssh client, X2Go

Windows:

Putty, MobaXterm

OpenOnDemand

Transfer files

Linux, Mac:

scp, sftp, rsync

Windows:

WinScp,
MobaXterm,
FileZilla, PuTTy

OpenOnDemand

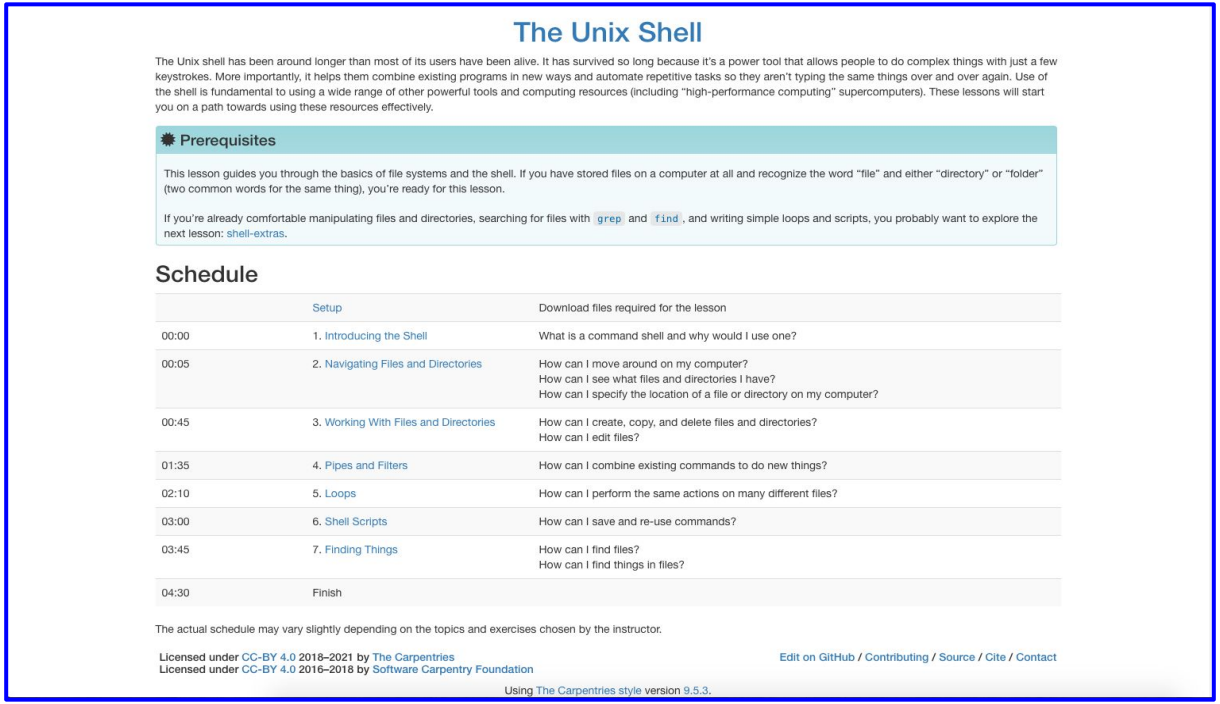
HPC work

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

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.

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

Carpentry courses for beginners:

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

https://westgrid.github.io/trainingMaterials/

University of Manitoba

Connect and 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?

Terminal:

~$ 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.westgrid.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**.

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.westgrid.ca
  - ✓ **Password:** your password
  - ✓ **Port:** 22
- **Use CygWin:** same environment as Linux

University of Manitoba

X2Go: Linux/Mac/Windows

Why X2Go: Access to GUI

How to use X2Go?

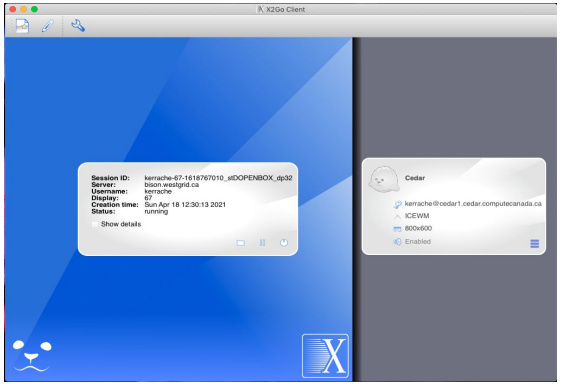
- Ask first if X2Go is installed on the remote machine.
- If yes, install X2Go client on your laptop or Desktop.
- Linux, Windows, Mac (XQuartz)
- Launch X2Go.
- Create a session and connect.

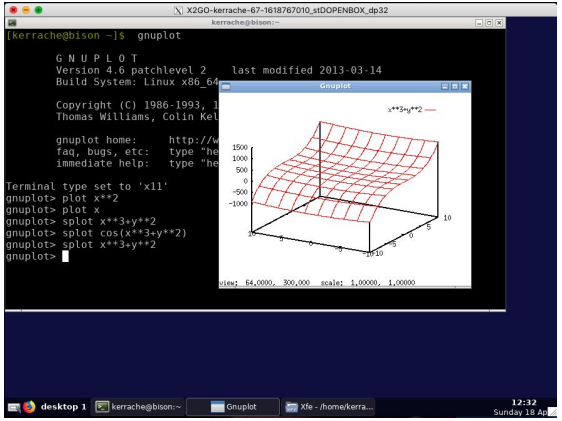
  Login: your user name

  Host: bison.westgrid.ca {or tatanka.westgrid.ca}

  Port: 22

  Session: ICEWM

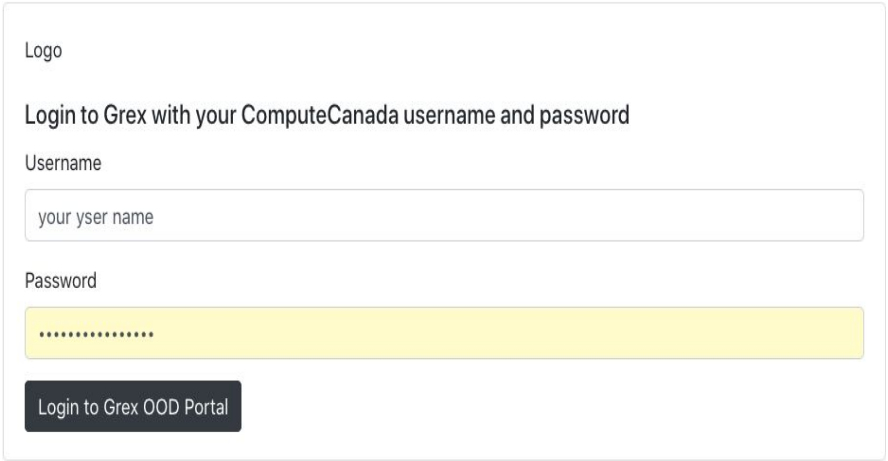


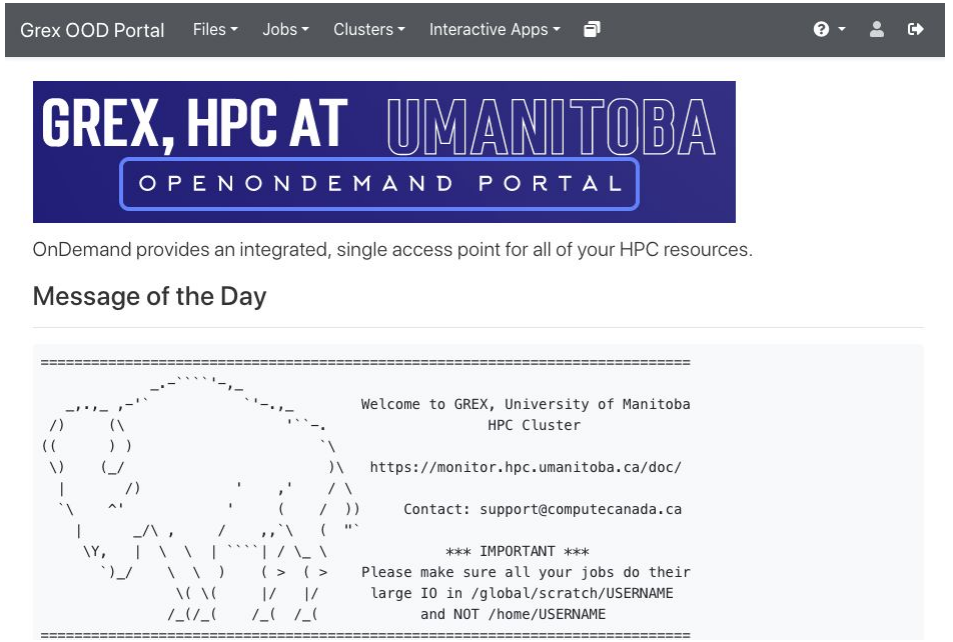


OOD: OpenOnDemand web portal

Connect to OOD using: [UManitoba VPN]:

- ★ Make sure Pulse Secure VPN is connected
- ★ Point your Web browser to https://aurochs.westgrid.ca
- ★ Use your Compute Canada username and password to log in to Grex OOD.





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

University of Manitoba

File system and quota

Alliance [Compute Canada]:

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

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

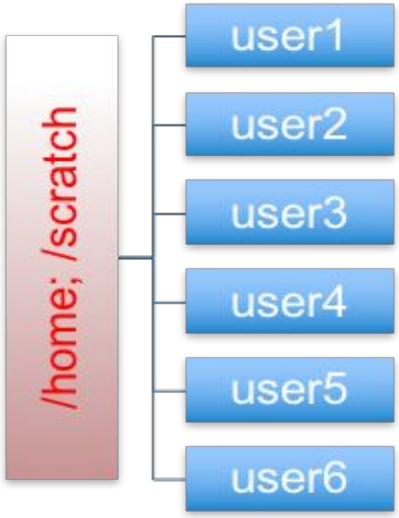
Grex:

/home/$USER:

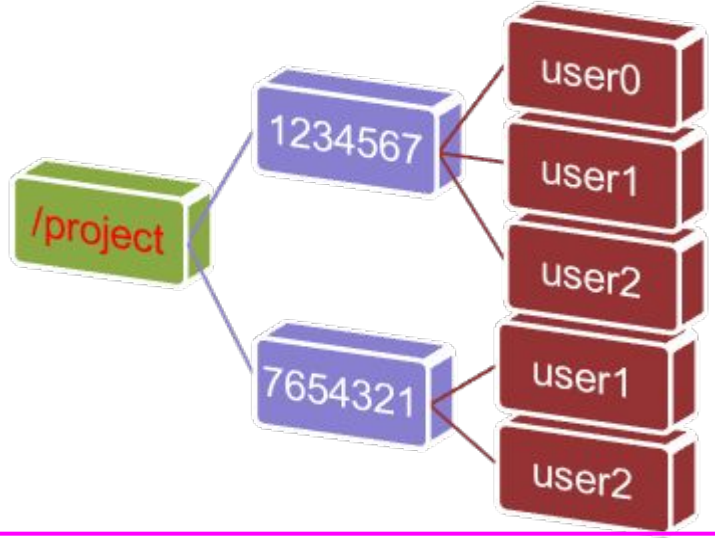
100 GB, backup

/global/scratch/$USER:

4 TB, no backup, no purge.



Project on the Alliance's clusters



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)	128G/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/500k

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 [some options] [Target] [Destination]

Syntax for rsync: rsync [some 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.

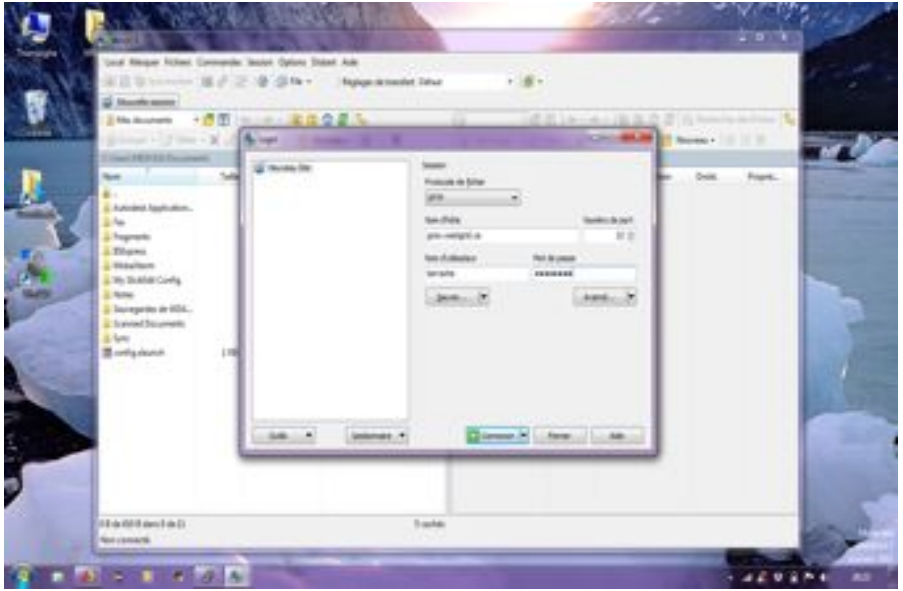
username@grex.westgrid.ca:/home/username/{Your_Dir}; ~/{Your_Dir}

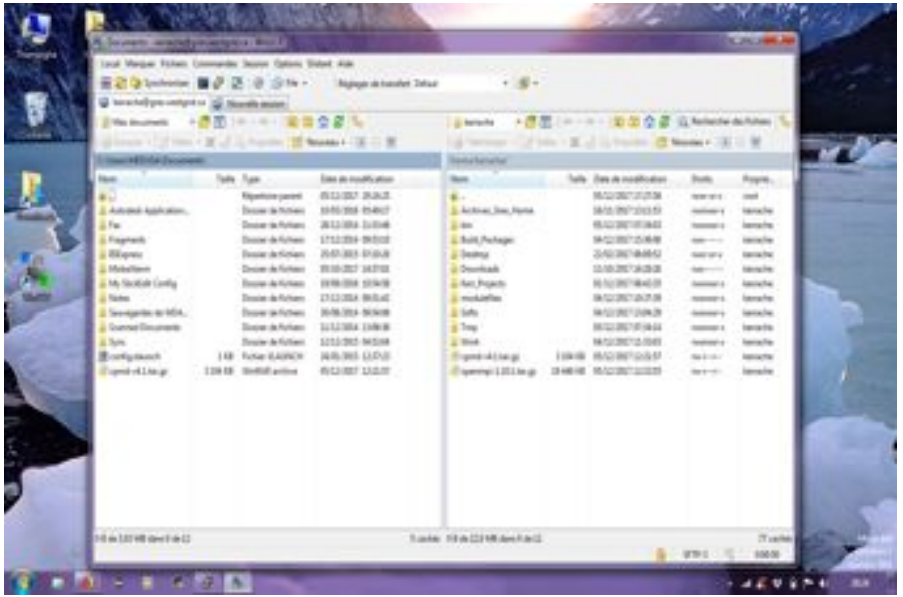
username@grex.westgrid.ca:/global/scratch/username/{Your_Dir}

[~@Mac]: scp -r TEST username@grex.westgrid.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

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, Tools, …

- ★ Commercial Software: restricted

  Contact support with some details about the license, …

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

University of Manitoba

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.computecanada.ca/wiki/Available_software
- ★ **Grex:**
  - **GrexEnv:** modules installed locally on Grex [about 500 modules].
  - **CCEnv:** access to public repository of Alliance [Compute Canada].

Find a software: Lmod

★ 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
```

University of Manitoba

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
```

University of Manitoba

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

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 test}
  - Performance can suffer greatly from over-subscription
- For interactive work, submit interactive jobs: salloc [some options]
  - SLURM uses salloc for interactive jobs
  - The jobs will run on dedicated compute nodes
- 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 [some 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

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 4944244
salloc: job 4944244 queued and waiting for resources
salloc: job 4944244 has been allocated resources
salloc: Granted job allocation 4944244
salloc: Waiting for resource configuration
salloc: Nodes n085 are ready for job
                                        Load modules + run tests
[someuser@n085 ]$  exit
exit
salloc: Relinquishing job allocation 4944244
```

```
--ntasks=1
--cpus-per-task=4
--mem-per-cpu=1000M
--time=1:00:00
--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 4944245
salloc: Waiting for resource configuration
salloc: Nodes n339 are ready for job
```

Load modules + run tests

```
[someuser@n339 ]$  exit
exit
salloc: Relinquishing job allocation 4944245
```

```
--nodes=1
--ntasks=1
--cpus-per-task=4
--mem-per-cpu=1000M
--mem=4000M
--time=3:00:00
--account=def-someprof
--partition=skylake
--x11
```

University of Manitoba

SLURM 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

Default parameters:

CPUs: 1

Time: 0-3:00

Memory: 256mb

University of Manitoba

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: E-mail-notification, … etc.

Submit and monitor the job:

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

Partition:

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

University of Manitoba

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)
#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:00

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=3-00:00:00
#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 -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=1500M
#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=1500M
#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
```

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`"
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
```

University of Manitoba

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

University of Manitoba

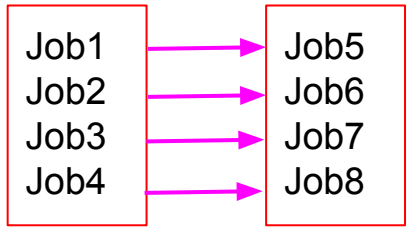
Bundle jobs with GLOST

```
#!/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 making and 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                                      # 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 --states=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

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

University of Manitoba

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 (Compute Canada 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

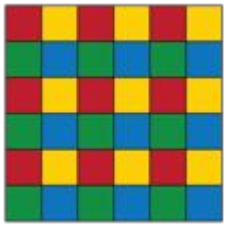
Optimization:
Better performance with 8 CPU
Memory: 4000 M per core [32 GB]

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

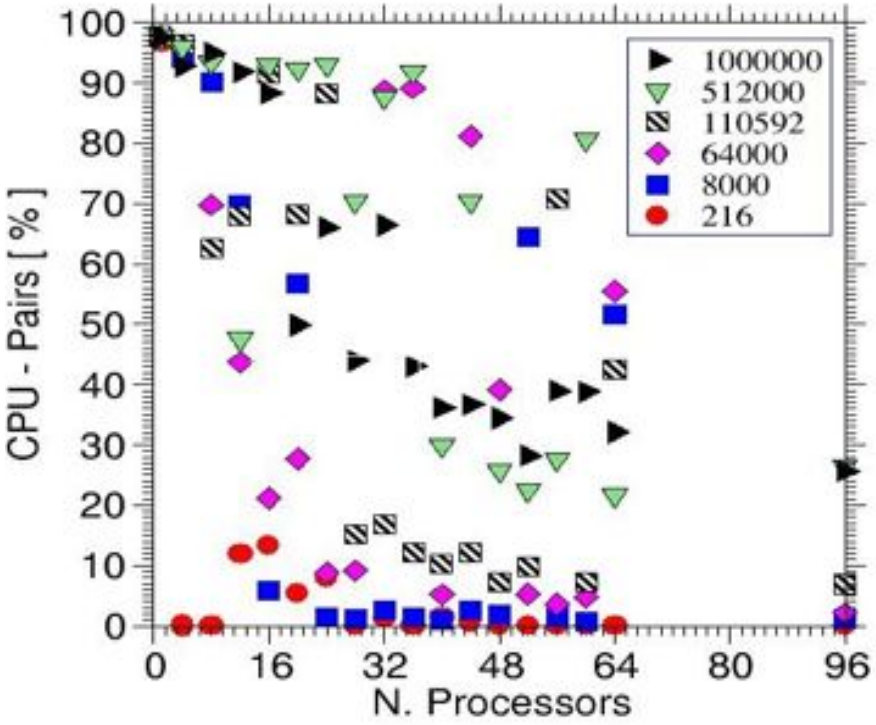
University of Manitoba

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 resource: benchmarks
- ★ Know time limits and estimate runtime of your jobs
  - comes after some trial and error, 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}.

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.

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
```

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.

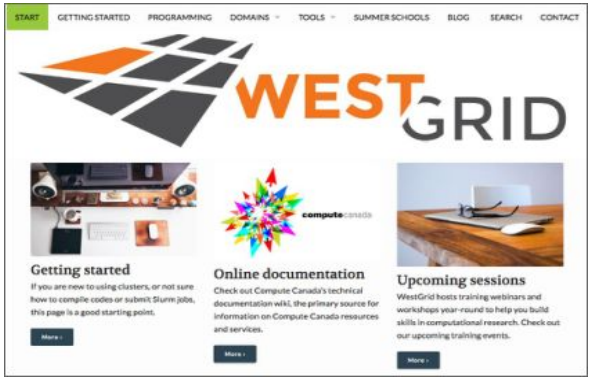
University of Manitoba

Thank you for your attention

Any question?

Useful links

The Alliance [former Compute Canada]:
https://docs.computecanada.ca/wiki/Compute_Canada_Documentation
CCDB: https://ccdb.computecanada.ca/security/login
CC Software: https://docs.computecanada.ca/wiki/Available_software
Running Jobs: https://docs.computecanada.ca/wiki/Running_jobs
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/



Help and support on CC: support@computecanada.ca
WG training material: https://westgrid.github.io/trainingMaterials/