

Introduction to HPC and available resources: UofM and Compute Canada

UofM-Spring-Workshop 2021
April 21st-22nd, 2021

Ali Kerrache



**University
of Manitoba**



★ Available resources:

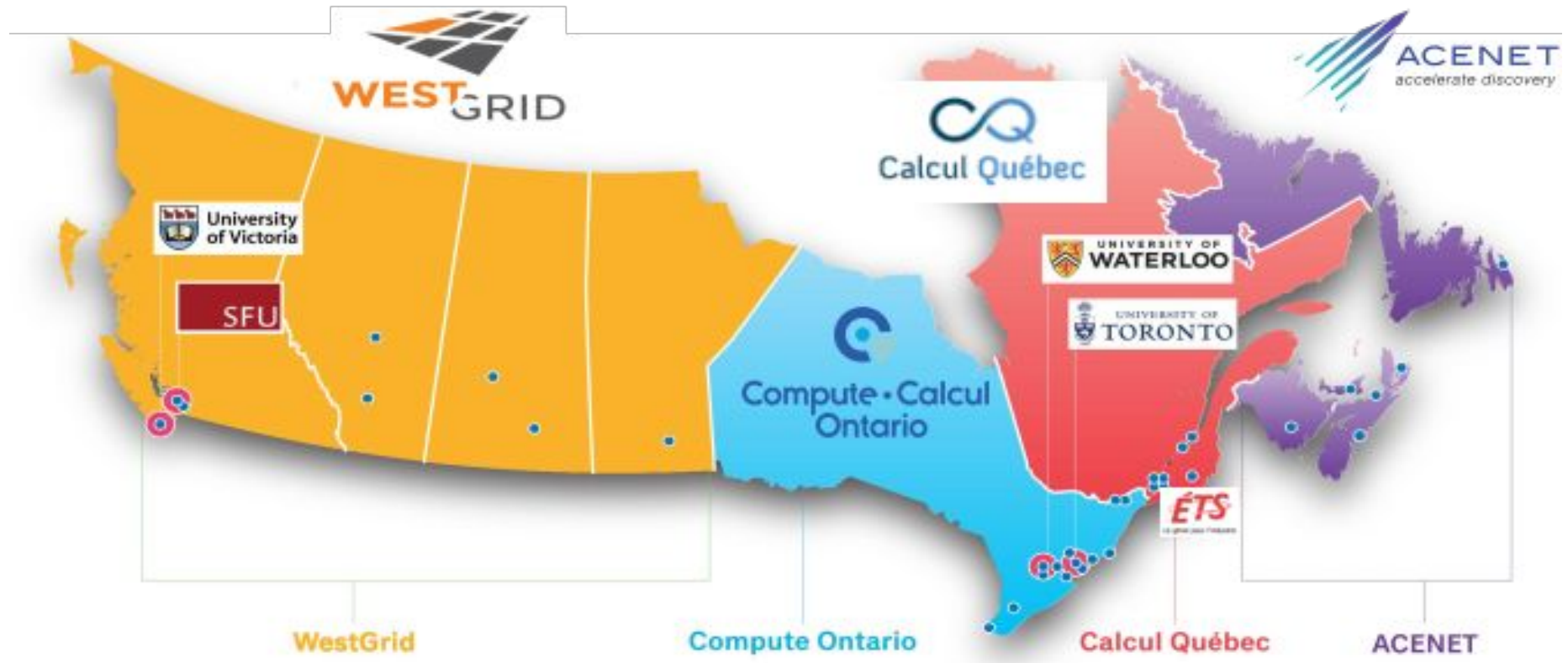
- Compute Canada: cedar, graham, beluga, niagara, cloud.
- Grex (UofM)
- New additions to Grex: **hardware** and **RAC-2021**

★ Basic tools for using HPC clusters:

- Linux shell (Terminal)
- Connect to a cluster: ssh client, PuTTY, MobXterm
- Transfer files: scp, sftp, WinSCP, ...
- Submit and monitor jobs: sbatch, salloc, squeue, ... etc.



Compute Canada partners



Compute Canada clusters

System	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
Niagara; Mist	80,640	216	16 PB	Large parallel jobs; [4 NVIDIA V100-32GB]
Arbutus	16008	108	17.3 PB	Physical cores: generally hyper-threaded.
GP cloud	*	*	*	Cloud partitions are available on GP systems for special purposes.

GreX, a cluster for UofM users

Partition	Nodes [CPUs/GPUs]	Cores/node	Cores	Memory	Max Wall Time
compute	288	12	3456	46 GB	21 days
bigmem	24	12	288	94 GB	14 days
skylake	12	40	480	376 GB	14 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
davis; -b	4	20	80	31GB	21 days / 7 days
livi, -b	1 [16 V100 - 32 GB]	48	48	1.5 TB	21 days / 7 days
*	42	52	2184	96 GB	*

Partitions on Grex: **partition-list**

```
alikerache — kerrache@tatanka:~ — ssh -YX kerrache@tatanka.westgrid.ca — 82x15
[kerrache@tatanka ~]$ partition-list
```

PARTITION	NODES(A/I)	TIMELIMIT	DEFAULTTIME	AVAIL	CPUS(A/I/O/T)	MEMORY
compute*	141/68	21-00:00:0	3:00:00	up	1511/997/948/3456	47000
bigmem	22/0	14-00:00:0	3:00:00	up	252/12/12/276	93000
gpu	0/2	3-00:00:00	3:00:00	up	0/64/0/64	191000
skylake	8/1	14-00:00:0	3:00:00	up	209/151/120/480	381500
test	10/0	21-00:00:0	3:00:00	up	120/0/24/144	47000
test1	8/1	21-00:00:0	3:00:00	up	209/151/40/400	381500
stamps	0/3	21-00:00:0	3:00:00	up	0/96/0/96	191000
stamps-b	0/3	7-00:00:00	3:00:00	up	0/96/0/96	191000
davis	0/4	21-00:00:0	3:00:00	up	0/80/0/80	31000
davis-b	0/4	7-00:00:00	3:00:00	up	0/80/0/80	31000
livi	0/1	21-00:00:0	3:00:00	up	0/48/0/48	1501000
livi-b	0/1	7-00:00:00	3:00:00	up	0/48/0/48	1501000

```
[kerrache@tatanka ~]$
```

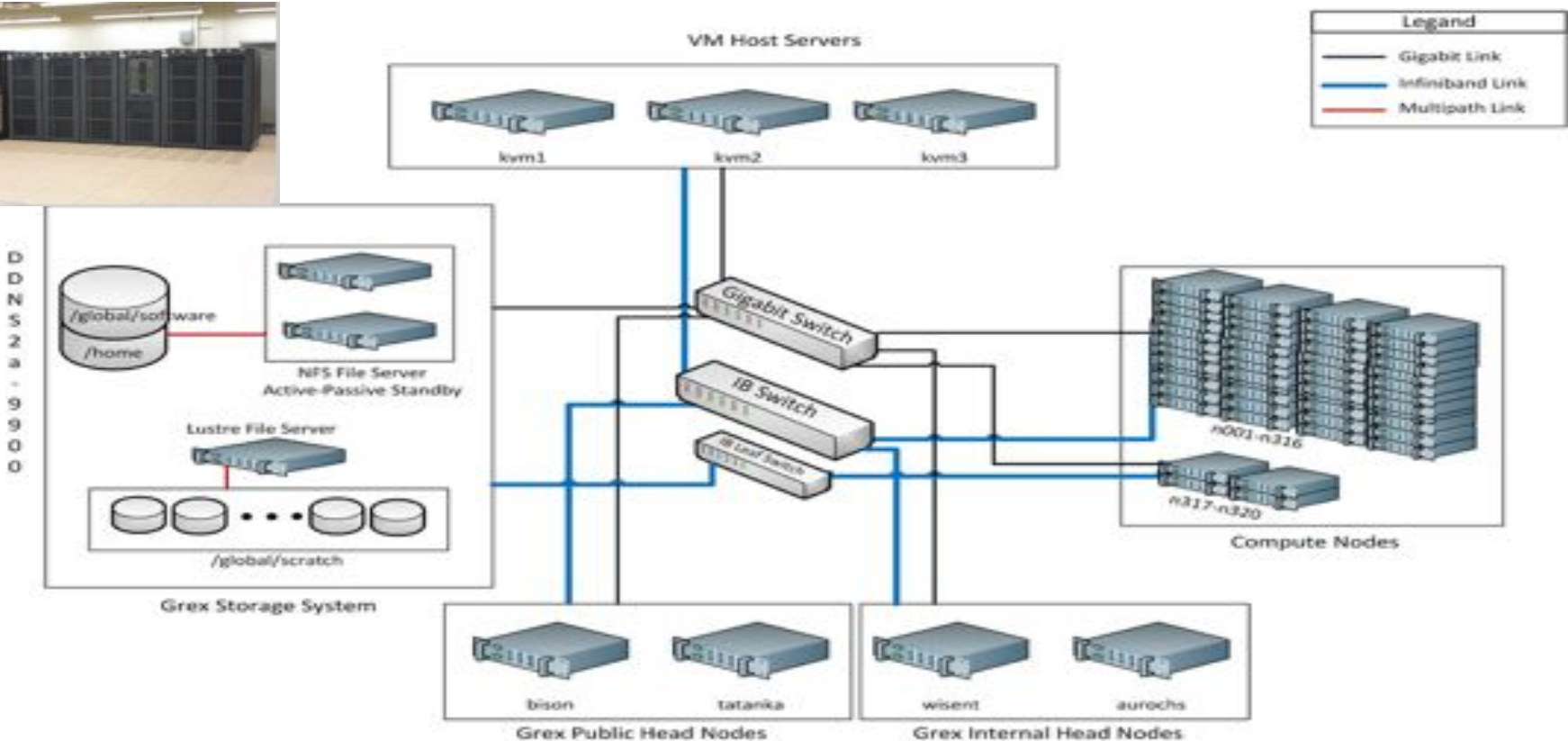
GreX upgrade and RAC 2021

- ★ **GreX upgrade: available for RAC 2021 (ARC@umanitoba.ca)**
 - 42 compute nodes [52 cores, 96 GB of memory]
- ★ **RAC 20: 2664 CPU cores (deadline: May 1st, 2021)**
 - 12 nodes [40 cores, *Intel CascadeLake* 6248 2.5GHz, 384 GB RAM]
 - 42 nodes [52 cores, *Intel CascadeLake* 6230R 2.1GHz, 96 GB RAM]
- ★ **GP resources: first arrived, first served**
 - 288 of 12 cores.
 - 24 of 12 cores.
 - 2 GPUs [32 cores, Tesla V100 32GB, 178 GB]
- ★ **Contributed hardware:**
 - partitions with suffix “-b”
 - could be used by any other user when not used by the contributor.





Structure of HPC clusters



Access to Compute Canada/Grex

Step 1:

Faculty member registers in the Compute Canada Database (CCDB): <http://ccdb.computecanada.ca>

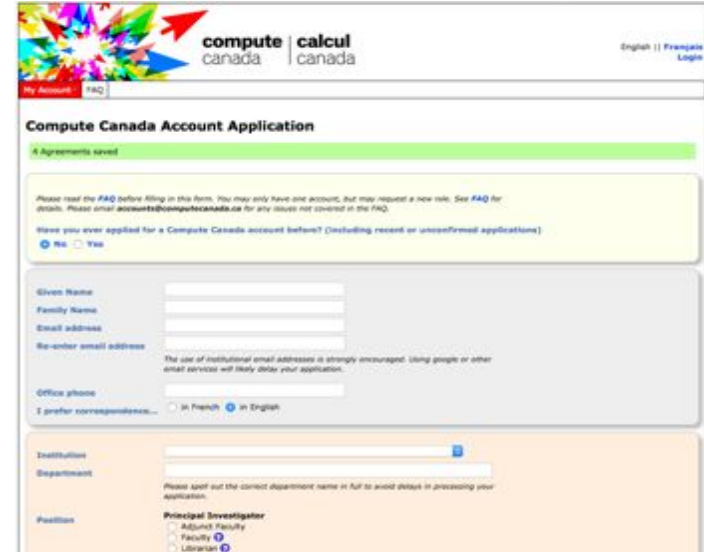
Step 2:

Once an account is approved, students / colleagues can register as group members (require CCRI):

CCDB account Gives access to the New systems:

Access to Compute Canada and Grex resources is free for the eligible researchers. Everyone gets a “Default” share. Every PI gets 1 TB of storage by default.

Resource Allocation Competitions (about 80%) are held annually, to distribute resources based on proposal’s merit. The remaining 20% are used for default share.



The screenshot shows the 'Compute Canada Account Application' form. At the top, there is a logo for 'compute canada' and 'calcul canada' with a colorful starburst graphic. Below the logo, there are links for 'English' and 'Français', and a 'Login' link. The main heading is 'Compute Canada Account Application'. A green bar indicates '4 Agreements saved'. Below this, there is a yellow box with instructions: 'Please read the FAQ before filling in this form. You may only have one account, but may request a new role. See FAQ for details. Please email accounts@computecanada.ca for any issues not covered in the FAQ.' A question asks 'Have you ever applied for a Compute Canada account before? (Including recent or unapproved applications)' with radio buttons for 'No' and 'Yes'. The form includes several input fields: 'Given Name', 'Family Name', 'Email address', 'Re-enter email address', and 'Office phone'. There is a note: 'The use of institutional email addresses is strongly encouraged. Using people or other email services will likely delay your application.' Below these fields, there is a language preference section: 'I prefer correspondence...' with radio buttons for 'In French' and 'In English'. The bottom section is for institutional information, including 'Institution', 'Department', and 'Position'. A note says: 'Please spell out the correct department name in full to avoid delay in processing your application.' There is a 'Principal Investigator' section with radio buttons for 'Adjunct Faculty', 'Faculty', and 'Librarian'.

Compute Canada: [Rapid Access Service](#); 10 TB of storage per cluster.
RAC for storage > 10 TB.
Send an email to: support@computecanada.ca

HPC: workflow and tools

Connect to a cluster

Linux:

ssh; X2Go

Mac:

ssh, X2Go

Windows:

Putty, MobaXterm, ...

Transfer files

Linux:

scp; sftp; rsync

Mac:

ssp, sftp; rsync; ...

Windows:

WinScp, FileZilla,
MobaXterm, ...

HPC

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



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.

Licensed under CC-BY 4.0 2018–2021 by The Carpentries
Licensed under CC-BY 4.0 2016–2018 by Software Carpentry Foundation

[Edit on GitHub](#) / [Contributing](#) / [Source](#) / [Cite](#) / [Contact](#)

Using The Carpentries style version 9.5.3.

Carpentry courses for beginners:

- **Introducing the shell**
- **Navigating with files and directories**
- **Working with files and directories**
- **Pipes and filters**
- **Loops**
- **Shell scripts**
- **Finding files and programs**

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



How to connect to a cluster?

Syntax:

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

options=-X; -Y {X11 forwarding}

Windows: install PuTTY, MobaXterm, ...

Mac: install XQuartz

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

❖ 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





Why X2Go: Access to GUI

How to use X2Go?

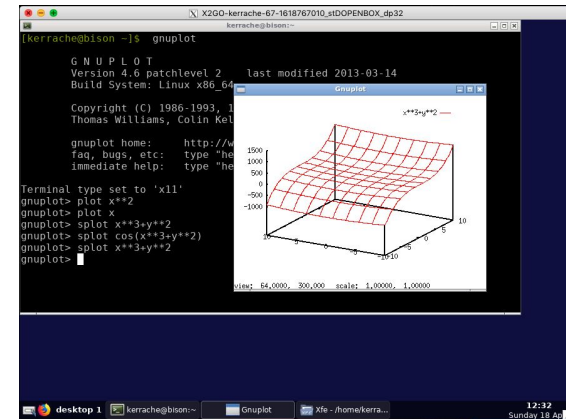
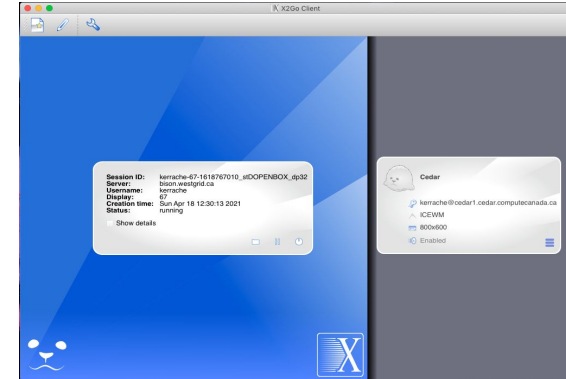
- Ask first if X2Go is installed on the remote machine.
- If yes, install X2Go client on your laptop or Desktop.
- Linux, Windows, Mac (XQuartz)
- Launch X2Go.
- Create a session and connect.

Login: your user name

Host: bison.westgrid.ca (or tatanka.westgrid.ca)

Port: 22

Session: ICEWM





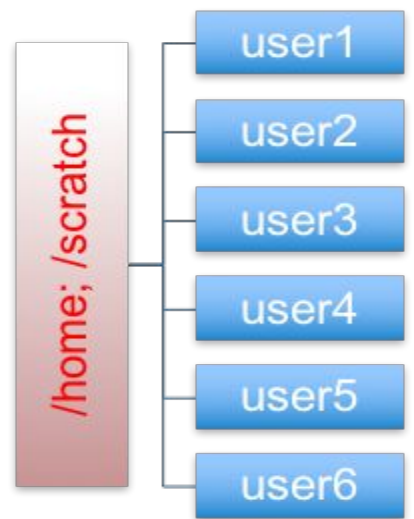
File system and quota

Compute Canada:

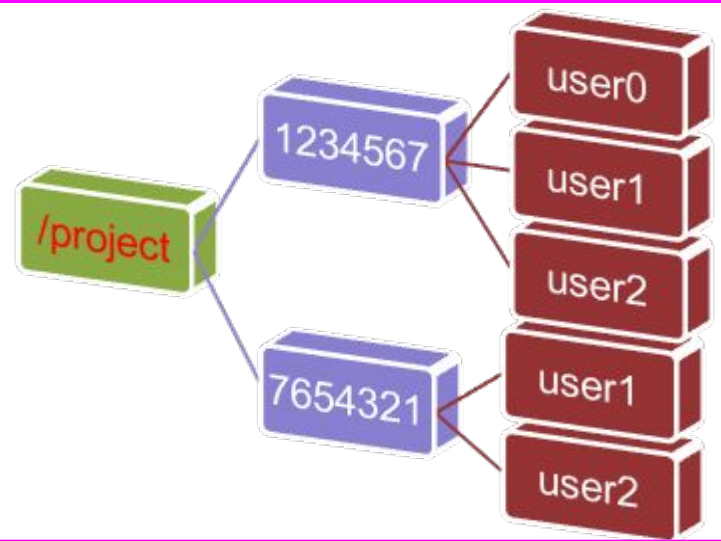
/home/\$USER: **50** GB, daily backup
/scratch/\$USER: **20** TB, no backup, purged

GreX:

/home/\$USER:
30 GB, backup
/global/scratch/\$USER:
2 TB, no backup, no purge.



Project in Compute Canada clusters



1 TB per group; extension up to **10 TB**
Backup; Allocatable via RAC (>10 TB)



Quota: **diskusage_report**

```
[kerrache@cedar1: ~]$ diskusage_report
```

Description	Space	# of files
/home (user kerrache)	72M/50G	652/500k
/scratch (user kerrache)	1978M/20T	8517/1000k
/project (group kerrache)	0/2048k	0/500k
/project (group def-kerrache-ab)	40G/1000G	327/500k
/project (group def-kerrache)	1838G/10T	9623/500k

```
[kerrache@tatanka ~]$ diskusage_report
```

Description (FS)	Space (U/Q)	# of files (U/Q)
/home (kerrache)	226M/31G	2381/500k
/global/scratch (kerrache)	519G/2147G	27k/1000k



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

Check if **scp**; **sftp**; **rsync** are supported.

Syntax for scp: scp [Options] [Target] [Destination]

Syntax for rsync: rsync [Options] [Target] [Destination]

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

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

Destination: where to copy the files (exact path).

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

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

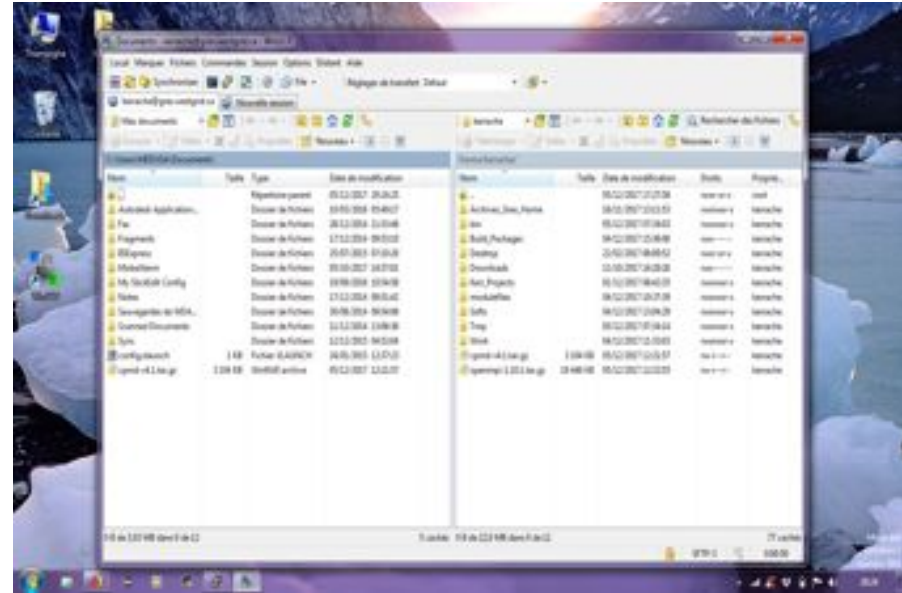
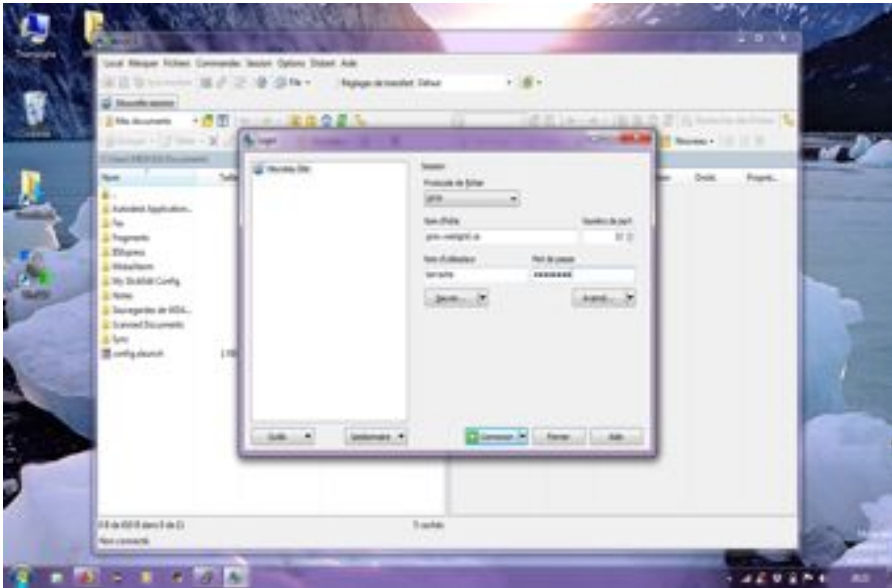
```
username@grex.westgrid.ca:~/{Your_Dir}
```

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



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).

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

Up to a user, ...

Free Software: GNU Public License.

Open Source, Binaries, Libraries, ...

Commercial Software:

Contact us with some details about the license, ...

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



★ 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

- ★ Number-crunching software environment:
 - Compilers (GCC, Intel), BLAS/LAPACK/PETSc, 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:**
 - **CCEnv:** access to public repository of Compute Canada
 - **GreXEnv:** modules installed locally on Grex.

Find a software on a cluster

★ 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



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 oversubscription
- ★ **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`
- ★ **For interactive work, submit interactive jobs:** salloc
 - SLURM uses salloc for interactive jobs
 - The jobs will run on dedicated compute nodes



- ★ 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 using?
 - 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



Example of SLURM script

```
#!/bin/bash
#SBATCH --account=def-somegroup
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=3-00:00:00
#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 myscript.sh`
- `queue -u $USER`

Partition:

- `partition-list`
- `sinfo -p <partition name>`



Monitor 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.



Custom script: grex-summarize-queue

```
[kerrache@bison ~]$ grex-summarize-queue
-----
grex-summarize-queue.py: queue summarized by user.
=====> Collected at: 2021-04-18 12:11:53.460199
-----

```

RUNNING-JOBS				PENDING-JOBS				USER	ACCOUNT
CORES	GPUS	JOBS	HOURS	CORES	GPUS	JOBS	HOURS	NAME	NAME
864	0	3	71	0	0	0	0	umbergm5	def-bcwang
544	0	43	431	0	0	0	0	zhang86	def-schrecke
200	0	2	6	8600	0	86	9	aleila	def-torabi
96	0	3	197	40	0	1	335	ismael	def-jhollett
48	0	2	57	0	0	0	0	sohail	def-schrecke
39	0	3	126	0	0	0	0	babarinv	def-telichev
36	0	1	111	0	0	0	0	rico	def-schrecke
24	0	1	170	0	0	0	0	xiaojw	def-dengc
16	0	2	242	0	0	0	0	singhs34	def-davisr1
12	0	1	487	0	0	0	0	umdorge6	def-bibeauel
4	0	1	196	0	0	0	0	bergmal6	rkm-871-aa
0	0	0	0	120	0	1	95	xwj	def-bcwang
TOTAL-RUNNING-JOBS				TOTAL-PENDING-JOBS				TOTAL	TOTAL
CORES	GPUS	JOBS	HOURS	CORES	GPUS	JOBS	HOURS	USERS	ACCOUNTS
1883	0	62	182	8760	0	88	12	12 total users	9 total accounts

```
[kerrache@bison ~]$
```



- ★ 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.



- ★ **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.



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**).

Thank you for your attention

Any question?



Compute Canada:

https://docs.compute canada.ca/wiki/Compute_Canada_Documentation

CCDB: <https://ccdb.compute canada.ca/security/login>

CC Software: https://docs.compute canada.ca/wiki/Available_software

Running Jobs: https://docs.compute canada.ca/wiki/Running_jobs

PuTTY: <http://www.putty.org/>

MobaXterm: <https://mobaxterm.mobatek.net/>

X2Go: <https://wiki.x2go.org/doku.php>

GreX: <https://monitor.hpc.umanitoba.ca/doc/>

Help and support on CC: support@compute canada.ca

WG training material: <https://westgrid.github.io/trainingMaterials/>

