Martine Tétreault's Lab – CRCHUM

A Guide to Bioinformatics Servers

All about Compute Canada and an Introduction to Linux

Marjorie Labrecque et Valerie Triassi
29/08/2022

Table des matières

Compute Canada

***** Compute Canada in now managed by the Digital Research Alliance of Canada *****

Accessing the remote server

Secure Shell (SSH) is a widely-used standard to connect to remote machines in a secure way. The entire SSH connection is encrypted - especially the login credentials (username and password). SSH is the normal way for Compute Canada users to connect in order to execute commands, submit jobs, follow the progress of these jobs and in some cases, transfer files.

- On **macOS** and **Linux** the most widely used client is OpenSSH, a command line application installed by default on these platforms.
- For recent versions of **Windows**, SSH is available in the PowerShell terminal, in the cmd prompt, or through Windows Subsystem for Linux (WSL). There are also some 3rd-party SSH clients that are popular, such as **PuTTY**, MobaXTerm, WinSCP, and Bitvise.

To use any of these implementations of SSH successfully, you need to know (1) the name of the machine to which you want to connect, (2) your username and (3) your password.

1. The **machine name** will be something like beluga.computecanada.ca
2. Your **username** is your Compute Canada default account (https://ccdb.computecanada.ca/security/login), typically something like jsmith, and the password is the same one you use to log in to the Compute Canada database, ccdb.computecanada.ca. The username is not your CCI, like abc-123, nor a CCRI like abc-123-01, nor your email address.

```
ssh username@machine_name
```

When your account is created on a Compute Canada cluster, your home directory will not be entirely empty. It will contain references to your scratch and project spaces through the mechanism of a symbolic link, a kind of shortcut that allows easy access to these other filesystems from your home directory. Note that these symbolic links may appear up to a few hours after you first connect to the cluster. While your home and scratch spaces are unique to you as an individual user, the project space is shared by a research group.

To see the project groups you may use the following command:

```
stat -c %G $HOME/projects/*/
```

In the Tetreault lab this command has the following result:

def-tetreaum (default group, limited space, **DO NOT USE**)

rrg-tetreaum (the lab's project group **you should use**)

Storage types

HOME: While your home directory may seem like the logical place to store all your files and do all your work, in general this isn't the case - your home normally has a relatively small quota and doesn't have especially good performance for the writing and reading of large amounts of data. The most logical use of your home directory is typically source code, small parameter files and job submission scripts.

PROJECT: The project space has a significantly larger quota and is well-adapted to sharing data among members of a research group since it, unlike the home or scratch, is linked to a professor's account rather than an individual user. The data stored in the project space should be fairly static, that is to say the data are not likely to be changed many times in a month. Otherwise, frequently changing data - including just moving and renaming directories - in project can become a heavy burden on the tape-based backup system.

SCRATCH: For intensive read/write operations on large files (> 100 MB per file), scratch is the best choice. Remember however that important files must be copied off scratch since they are not backed up there, and older files are subject to purging. The scratch storage should therefore be used for temporary files: checkpoint files, output from jobs and other data that can easily be recreated.

SLURM_TMPDIR: While a job is running, $SLURM_TMPDIR is a unique path to a temporary folder on a local fast filesystem on each compute node reserved for the job. This is the best location to temporarily store large collections of small files (< 1 MB per file). Note: this space is shared between jobs on each node, and the total available space depends on the node specifications. Finally, when the job ends, this folder is deleted.

Disk quota exceeded error

Some users have seen this message or some similar quota error on their /project folders. Other users have reported obscure failures while transferring files into their /project folder from another cluster. Many of the problems reported are due to bad file ownership.

Use `diskusage_report` to see if you are at or over your quota:

```
[ymartin@cedar5 ~]$ diskusage_report
                             Description                Space           # of files
                 Home (user ymartin)                345M/50G            9518/500k
              Scratch (user ymartin)                 93M/20T            6532/1000k
             Project (group ymartin)              5472k/2048k             158/5000k
         Project (group def-zrichard)                20k/1000G               4/5000k
```

The example above illustrates a frequent problem: /project for user ymartin contains too much data in files belonging to group ymartin. The data should instead be in files belonging to def-zrichard.

Note the two lines labelled Project.

- Project (group ymartin) describes files belonging to group ymartin, which has the same name as the user. This user is the only member of this group, which has a very small quota (2048k).

- Project (group def-zrichard) describes files belonging to a project group. Your account may be associated with one or more project groups, and they will typically have names like def-zrichard, rrg-someprof-ab, or rpp-someprof.

In this example, files have somehow been created belonging to group ymartin instead of group def-zrichard. This is neither the desired nor the expected behaviour.

By design, new files and directories in /project will normally be created belonging to a project group. The main reasons why files may be associated with the wrong group are

- files were moved from /home to /project with the mv command; to avoid this, see section about transferring data
- files were transferred from another cluster using rsync or scp with an option to preserve the original group ownership. If you have a recurring problem with ownership, check the options you are using with your file transfer program;
- you have no setgid bit set on your Project folders.

How to fix the problem

If you already have data in your /project directory (or any other directory, just change the path) with the wrong group ownership, you can use the find to display those files:

```
lfs find ~/projects/*/ -group $USER
```

Next, change group ownership from $USER to the project group, for example:

```
chown -h -R $USER:def-professor -- ~/projects/def-professor/$USER/
```

Then, set the setgid bit on all directories (for more information, see Group ID) to ensure that newly created files will inherit the directory's group membership, for example:

```
lfs find ~/projects/def-professor/$USER -type d -print0 | xargs -0 chmod g+s
```

Finally, verify that project space directories have correct permissions set:

```
chmod 2770 ~/projects/def-professor/

chmod 2700 ~/projects/def-professor/$USER
```

For other Frequently Asked Questions:

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

Best practices

- Regularly clean up your data in the scratch and project spaces, because those filesystems are used for huge data collections.
- Only use text format for files that are smaller than a few megabytes.
- As far as possible, use scratch and local storage for temporary files. For local storage you can use the temporary directory created by the job scheduler for this, named $SLURM_TMPDIR.
- If your program must search within a file, it is fastest to do it by first reading it completely before searching.
- If you no longer use certain files but they must be retained, archive and compress them, and if possible move them to an alternative location like nearline.

Finding folders with lots of files (in current directory):

```
for FOLDER in $(find . -maxdepth 1 -type d | tail -n +2); do
  echo -ne "$FOLDER:\t"
  find $FOLDER -type f | wc -l
done
```

Finding folders using the most disk space (in current directory):

```
du -sh  * | sort -hr | head -10
```

Linux introduction

Following your connection, you are directed to your $HOME directory (the UNIX word for "folder") for your user account. When your account is created, your $HOME only contains a few hidden configuration files that start with a ".", and nothing else.

On a Linux system, you are **strongly discouraged to create files or directories that contain names with spaces or special characters, including accents.**

Text file line endings

For historical reasons, Windows and most other operating systems, including Linux and OS X, disagree on the convention that is used to denote the end of a line in a plain text ASCII file. Text files prepared in a Windows environment will therefore have an additional invisible "carriage return" character at the end of each line and this can cause certain problems when reading this file in a Linux environment. For this reason you should either consider creating and editing your text files on the cluster itself using standard Linux text editors like emacs, vim and nano or, if you prefer Windows, to then use the command **dos2unix <filename>** on the cluster login node to convert the line endings of your file to the appropriate convention.

Copy & Paste

In the Linux terminal you can't use [Ctrl]+C to copy text, because [Ctrl]+C means "Cancel" or "Interrupt" and stops the running program.

Instead you can use [Ctrl]+[Insert] to copy and [Shift]+[Insert] to paste in most cases under Windows and Linux, depending on your terminal program. Users of macOS can continue to use [Cmd]+C and [Cmd]+V to copy and paste.

Depending which terminal software you are using, you simply need to select the text to copy it into the clipboard, and you can paste from the clipboard by using either the right-click or middle-click (the default setting can vary).

Listing directory contents

To list all files in a directory in a terminal, use the ls (list) command:

`ls`	`-a`	To include hidden files
	`-t`	To sort results by date (from newest to oldest) instead of alphabetically
	`-l`	To obtain detailed information on all files (permissions, owner, group, size and last modification date)
	`-h`	Gives the file sizes in human readable format

Navigating the filesystem

To move about in the filesystem, use the cd command (change directory).

`cd`	`my_directory`	To change to `my_directory`
	`..`	To change to the parent folder
		To move back to your home directory ($HOME)

Creating and removing directories

To create (make) a directory, use the mkdir command:

```
mkdir my_directory
```

To remove a directory, use the rmdir command (only works if it is empty):

```
rmdir my_directory
```

Deleting files or directories

You can remove files using the rm command **(BE CAREFUL!!!)**:

`rm`	`my_file`	Remove a file
	`-r my_directory`	Recursively remove a directory

Copying and renaming files or directories

To copy a file use the cp command:

```
cp source_file destination_file
```

To recursively copy a directory:

```
cp -R source_directory destination_directory
```

To rename a file or a folder (directory), use the mv command (move):

```
mv source_file destination_file

mv source_directory destionation_directory
```

Viewing and editing files

To view a file read-only, use the less command:

```
less file_to_view
```

You can then use the arrow keys or the mouse wheel to navigate the document. You can search for something in the document by typing **/what_to_search_for**. You can quit less by pressing the q key.

The diff command allows you to compare two files:

```
diff file1 file2
```

The -y option shows both files side by side.

The grep command allows you to look for a given expression in one or multiple files.

```
grep 'tata' file1

grep 'tata' fil*
```

Note that, in Linux, the **"*" wildcard matches zero or more characters. The "?" wilcard matches exactly one character**.

The text to be searched for can also be variable. For example, to look for the text "number 10 » or "number 11", etc. with any number between 10 and 29, the following command can be used:

```
grep 'number [1-2][0-9]' file
```

List of all Regex : https://www.cyberciti.biz/faq/grep-regular-expressions/

Getting help

Generally, UNIX commands are documented in the reference manuals that are available on the servers. To access those from a terminal:

```
man command
```

man uses less, and you must **press q to exit this program**.

To find manual pages pertaining to a certain subject or keyword (for example "sujet"), please enter:

```
apropos sujet
```

By convention, the executables themselves contain some help on how to use them. Generally, you invoke this help using the command line argument -h or --help, or in certain cases, -help.

File permissions

UNIX systems support 3 types of permissions : **read (r)**, **write (w)** and **execute (x)**.

- For **files**, a file should be readable to be read, writable to be modified, and executable to be run (if it's a binary executable or a script).
- For a **directory**, read permissions are necessary to list its contents, write permissions enable modification (adding or removing a file) and execute permissions enable changing to it.

Permissions apply to 3 different classes of users, the **owner (u)**, the **group (g)**, and all others or **"the world" (o)**. To know which permissions are associated to files and subdirectories of the current directory, use the following command: ls -la

The 10 characters at the beginning of each line show the permissions.

The first character indicates the file type :

- -: a normal file
- d: a directory
- l: a symbolic link

Then, from left to right, this command shows read, write and execute permissions of the owner, the group and other users. Here are some examples :

- drwxrwxrwx: a world-readable and world-writable directory
- drwxr-xr-x: a directory that can be listed by everybody, but only the owner can add or remove files
- -rwxr-xr-x: a world-readable and world-executable file that can only be changed by its owner
- -rw-r--r--: a world-readable file that can only be changed by its owner.
- -rw-rw----: a file that can be read and changed by its owner and by its group
- -rw-------: a file that can only be read and changed by its owner
- drwx--x--x: a directory that can only be listed or modified by its owner, but all others can still pass it on their way to a deeper subdirectory
- drwx-wx-wx: a directory that everybody can enter and modify but where only the owner can list its contents

Important note: to be able to read or write in a directory, you need to have **execute permissions (x) set in all parent directories, all the way up to the filesystem's root (/)**. So if your home directory has drwx------ permissions and contains a subdirectory with drwxr-xr-x permissions, other users cannot read the contents of this subdirectory because they do not have access (by the executable bit) to its parent directory.

After listing the permissions, ls -la command gives a number, followed by the file owner's name, the file group's name, its size, last modification date, and name.

Permission number file owner's name file group's name size last modification date name

-rw-rw-r--. 1 mlab rrg-tetreaum 634 Jul 27 17:46 mapped_stats.txt

The **chmod command** allows you to change file permissions. The simple way to use it is to specify which permissions you wish to add or remove to which type of user. To do this, you specify the list of users (**u for the owner, g for the group, o for other users, a for all**), followed by a **+ to add permissions** or **- to remove permissions**, which is then followed by a list of permissions to modify (**r for read, w for write, x for execute**). Non-specified permissions are not affected. Here are a few examples:

Prevent group members and all others to read or modify the file secret.txt:

```
chmod go-rwx secret.txt
```

Allow everybody to read the file public.txt:

```
chmod a+r public.txt
```

Make the file script.sh executable:

```
chmod a+x script.sh
```

Allow group members to read and write in the directory shared:

```
chmod g+rwx shared
```

Prevent other users to read or modify your home directory:

```
chmod go-rw ~
```

It's also common for people to use "**octal notation**" when referring to Unix filesystem permissions even if this is somewhat less intuitive than the above symbolic notation. In this case, we use three bits to represent the permissions for each category of user, with these three bits then interpreted as a number from 0 to 7 using the formula **(read_bit)*4 + (write_bit)*2 + (execute_bit)*1**.

Note that to be able to exercise your rights on a file, you also need to be able to access the directory in which it resides. This means having both read and execute permission ("5" or "7" in octal notation) on the directory in question.

You can alter these permissions using the command chmod in conjunction with the octal notation discussed above, so for example:

```
chmod 770 name_of_file
```

The Sticky Bit

When dealing with a **shared directory where multiple users have read, write and execute permission**, as would be common in the project space for a professor with several active students and collaborators, **the issue of ensuring that an individual cannot delete the files or directories of another can arise**. For preventing this kind of behaviour the Unix filesystem developed the concept of the sticky bit by means of which the filesystem permissions for a directory can be restricted so that a file in that directory **can only be renamed or deleted by the file's owner or the directory's owner**. Without this sticky bit, users with write and execute permission for that directory can rename or delete any files that it may contain even if they are not the file's owner. The sticky bit can be set using the command chmod, for example

```
chmod +t <directory name>
```

or if you prefer to use the octal notation discussed above by using the mode 1000, hence

```
chmod 1774 <directory name>
```

to set the sticky bit and rwxrwxr-- permissions on the directory.

The sticky bit is represented in ls -l output by the **letter "t" or "T"** in the last place of the permissions field, like so:

```
ls -ld directory
```

output : drwxrws--**T** 2 someuser def-someuser 4096 Sep 25 11:25 directory

The sticky bit can be unset by the command

```
chmod -t <directory name>

chmod 0774 <directory name>
```

Set Group ID bit

When creating files and directories within a parent directory it is often useful to match the group ownership of the new files or directories to the parent directory's owner or group automatically. **If the setGID bit is enabled for a directory, new files and directories in that directory will be created with the same group ownership as the directory**. To illustrate the use of this mode, let us walk through an example.

Start by checking the groups that `someuser` belongs to with the `groups` command.

We want a newly created file to belong to the same group as the parent folder. Enable the setGID permission on the parent directory like so:

```
chmod g+s dirTest
```

Notice that the **x permission on the group permissions has changed to an s**. Now newly created files in dirTest will have the same group as the parent directory.

The **uppercase S indicates that execute permissions have been removed from the directory but the setGID is still in place**. It can be easy to miss this and may result in unexpected problems, such as others in the group not being able to access files within your directory.

Default filesystem permissions

Default filesystem permissions are defined by something called the umask. There is a default value that is defined on any Linux system. To display the current value in your session, you can run the command

```
umask -S            #If output is u=rwx,g=rx,o=
```

This means that, by default, new files that you create can be read, written and executed by yourself, they can be read and executed by members of the group of the file, and they cannot be accessed by other people. **The umask only applies to new files. Changing the umask does not change the access permissions of existing files.**

There may be reasons to define default permissions more permissive (for example, to allow other people to read and execute files), or more restrictive (not allowing your group to read or execute files). Setting your own umask can be done either in a **session, or in your .bashrc file**, by calling the command

```
umask <value>
```

`umask` value	`umask` meaning	Human-readable explanation
077	u=rwx,g=,o=	Files are readable, writable and executable by the owner only
027	u=rwx,g=rx,o=	Files are readable and executable by the owner and the group, but writable only by the owner
007	u=rwx,g=rwx,o=	Files are readable, writable and executable by the owner and the group
022	u=rwx,g=rx,o=rx	Files are readable and executable by everyone, but writable only by the owner
002	u=rwx,g=rwx,o=rx	Files are readable and executable by everyone, but writable only by the owner and the group

Default umask = 027

Transferring data

Beluga:

- Availability : March, 2019
- Login node : beluga.computecanada.ca
- Globus Endpoint : computecanada#beluga-dtn
- Data Transfer Node (rsync, scp, sftp,...) : **beluga.computecanada.ca**
- Béluga is a general purpose cluster designed for a variety of workloads and situated at the École de technologie supérieure in Montreal. The cluster is named in honour of the St. Lawrence River's Beluga whale population.

To and from your personal computer

You will need software that supports secure transfer of files between your computer and our machines. The commands scp and sftp can be used in a command-line environment on Linux or Mac OS X computers. On Microsoft Windows platforms, MobaXterm offers both a graphical file transfer function and a command-line interface via SSH, while WinSCP is another free program that supports file transfer. Setting up a connection to a machine using SSH keys with WinSCP can be done by following the steps in this link. PuTTY comes with pscp and psftp which are essentially the same as the Linux and Mac command line programs.

Important note: If it takes more than one minute to move your files to or from our servers, we recommend you install and try **Globus Personal Connect**. Globus transfers can be set up and will go on in the background without you.

Between resources

Globus is the preferred tool for transferring data between systems, and if it can be used, it should.

However, other common tools can also be found for transferring data both inside and outside of our systems, including

- SFTP
- SCP or Secure Copy
- rsync

Example:

```
scp file
someuser@beluga.computecanada.ca:/scratch/g/group/someuser/

rsync -rltv LOCALNAME
someuser@beluga.computecanada.ca:projects/rrg-professor/someuser/
```

where `LOCALNAME` can be a folder or file. For large transfers consider adding `--partial` so interrupted transfers maybe restarted and/or `--progress` to see a summary of the transfer progress.

From the World Wide Web

The standard tool for downloading data from websites is wget. Also available is curl. The two are compared in this StackExchange article (https://unix.stackexchange.com/questions/47434/what-is-the-difference-between-curl-and-wget).

Synchronizing files

To synchronize or "sync" files (or directories) stored in two different locations means to ensure that the two copies are the same. Here are several different ways to do this.

Globus

We find Globus usually gives the best performance and reliability.

Normally when a Globus transfer is initiated it will overwrite the files on the destination with the files from the source, which means all of the files on the source will be transferred. If some of the files may already exist on the destination and need not be transferred if they match, you should go to the bottom of the transfer window as shown in the screenshot and choose to "sync" instead.

Their checksums are different	This is the slowest option but most accurate. This will catch changes or errors that result in the same size of file, but with different contents.
File doesn't exist on destination	This will only transfer files that have been created since the last sync. Useful if you are incrementally creating files.
File size is different	A quick test. If the file size has changed then its contents must have changed, and it will be re-transferred.
Modification time is newer	This will check the file's recorded modification time and only transfer the file if it is newer on the source than the destination. If you want to depend on this, it is important to check the "preserve source file modification times" option when initiating a Globus transfer.

Using checksums to check if files match

If Globus is unavailable between the two systems being synchronized and Rsync is taking too long, then you can use a checksum utility on both systems to determine if the files match. In this example we use sha1sum.

```
find /home/username/ -type f -print0 | xargs -0 sha1sum | tee checksum-result.log
```

This command will create a new file called `checksum-result.log` in the current directory; the file will contain all of the checksums for the files in `/home/username/`. It will also print

out all of the checksums to the screen as it goes. If you have a lot of files or very large files you may want to **run this command in the background, in a screen or tmux session**; anything that allows it to continue if your SSH connection times out.

After you run it on both systems you can use the diff utility to find files that don't match.

```
diff checksum-result-silo.log checksum-dtn.log
```

It is possible that the `find` command will crawl through the directories in a different order, resulting in a lot of false differences so you may need to run sort on both files before running diff such as:

```
sort -k2 checksum-result-silo.log -o checksum-result-silo.log

sort -k2 checksum-dtn.log -o checksum-dtn.log
```

Clean Up

It is a good practice to look at your files regularly and see what can be deleted, but unfortunately many of us do not have the habit. A major data migration is a good reminder to clean up your files and directories. Moving less data will take less time, and storage space even on new systems is in great demand and should not be wasted.

- If you compile programs and keep source code, delete any intermediate files. One or more of make clean, make realclean, or rm *.o might be appropriate, depending on your makefile.
- If you find any large files named like core.12345 and you don't know that they are, they are probably core dumps and can be deleted.

Archive and compress

Most file transfer programs move one file of a reasonable size more efficiently than thousands of small files of equal total size. If you have directories or directory trees containing many small files, use tar to combine (archive) them.

Large files can benefit from compression in some cases, especially text files which can usually be compressed a great deal. Compressing a file only for the purpose of transferring it, and then decompressing it at the end of the transfer, will not necessarily save time though. It depends on how much the file can be compressed, how long it takes to compress it, and the transfer bandwidth. The calculation is described under "Data Compression and transfer discussion" in this document from the US National Center for Supercomputing Applications (https://bluewaters.ncsa.illinois.edu/data-transfer-doc).

If you decide compression is worthwhile you can again use tar for this, or gzip.

Avoid duplication

Try not to move the same data twice. If you are migrating from more than one existing system to one new system and you have data duplicated on the sources, choose one and only move the duplicate data from that one.

Beware of files with duplicate names, but which do not contain duplicate information. Ensure that you will not accidentally overwrite one file with another of the same name.

Migration of a large volume of data

If it is supported at your source site, use Globus Online to set up your file transfer. It is the most user-friendly and efficient tool we know for this task. Globus is designed to recover from network interruptions automatically. We recommend you select the following options at the bottom of the "Transfer files" screen:

- preserve source file modification times
- verify file integrity after transfer

If Globus is not supported at your source site, then the advice to compress data and avoid duplication is even more important. If you must use one of scp, sftp, or rsync, then:

- Make a schedule to migrate your data in blocks of a few hundred GBs at a time. If the transfer stops for some reason, you will be able to try again starting from the incomplete file, but you will not have to re-transfer files that are already complete. An organized list of files will help here.
- Check regularly to see that the transfer process has not stopped. File size is a good indicator of progress. If no files have changed size for several minutes, then something may have gone wrong. If restarting the transfer does not work, contact our Technical support.

Be patient. Even with Globus, transferring large volumes of data can be time consuming. Specific transfer speeds will vary a lot, but expect hundreds of gigabytes to take hours and terabytes to take days.

Archiving / Compressing

Archiving means **creating one file that contains a number of smaller files within it**. Reducing the number of files by creating an archive can improve the efficiency of file storage and help you stay within quota limits. Archiving can also improve the efficiency of file transfers. It is faster for the secure copy protocol (scp), for example, to transfer one archive file of a reasonable size than thousands of small files of equal total size.

Compressing means **encoding a file such that the same information is contained in fewer bytes of storage**. The advantage for long-term data storage should be obvious. For data transfers, the time spent for compressing data must be balanced against the time saved moving fewer bytes as described in this discussion of data compression and transfer from the US National Center for Supercomputing Applications.

- The best-known tool for archiving files in the Linux community is tar. Here is a tutorial on 'tar' (https://docs.alliancecan.ca/wiki/A_tutorial_on_%27tar%27).
- A replacement for tar called dar offers some advantages in functionality. Here is a tutorial on 'dar' (https://docs.alliancecan.ca/wiki/Dar). Both tar and dar can compress files as well as archive.
- The zip utility, more commonly used in the Windows community but available on our clusters, also provides both archiving and compression.
- Compression tools gzip, bzip2 and xz can be used in conjunction with tar, or by themselves.

SQLite

The SQLite software allows for the use of a **relational database which resides entirely in a single file stored on disk, without the need for a database server**. The data located in the file can be accessed using standard SQL (Structured Query Language) commands such as SELECT and there are APIs for several common programming languages. Using these APIs you can then interact with your SQLite database inside of a program written in C/C++, Python, R, Java and Perl. Modern relational databases contain datatypes for handling the storage of binary blobs, such as the contents of an image file, so storing a collection of 5 or 10 million small PNG or JPEG images inside of a single SQLite file may be much more practical than storing them as individual files. There is the overhead of creating the SQLite database and this approach assumes that you are familiar with SQL and designing a simple relational database with a small number of tables. Note as well that the performance of SQLite can start to degrade for very large database files, several gigabytes or more, in which case you may need to contemplate the use of a more traditional client/server database using MySQL or PostgreSQL.

Client/server SQL database engines strive to implement a shared repository of enterprise data. They emphasize scalability, concurrency, centralization, and control. SQLite strives to provide local data storage for individual applications and devices. **SQLite emphasizes economy, efficiency, reliability, independence, and simplicity.**

The SQLite executable is called sqlite3. It is available via the nixpkgs module, which is loaded by default on Compute Canada systems.

```
sqlite3 foo.sqlite
```

If the file `foo.sqlite` does not already exist, SQLite will create it and the client will start in an empty database, otherwise you will be connected to the existing database. You may then execute whichever queries you wish on the database, such as `SELECT * FROM tablename`; to print to the screen the entire contents of the table `tablename`.

Accessing SQLite from software

The most common way to interact with an SQLite (or other) database is through function calls to open a connection to the database; execute queries that can read, insert or update existing data; and close the connection to the SQLite database so that any changes are flushed to the SQLite file. In the simple example below, we suppose that the database has already been created with a table called `employee` that has two columns: the string `name` and the integer `age`.

```
#!/usr/bin/env python3


# For Python we can use the module sqlite3, installed in a
virtual environment,

# to access an SQLite database

import sqlite3


age = 34


# Connect to the database...

dbase = sqlite3.connect("foo.sqlite")


dbase.execute("INSERT INTO employee(name,age) VALUES(\"John
Smith\"," + str(age) + ");")


# Close the database connection

dbase.close()
```

Limitation

As its name suggests, SQLite is easy to use and intended for relatively simple databases which are neither excessively large (hundreds of gigabytes or more) nor too complicated in terms of their entity-relationship diagram (https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model).

As your SQLite database grows in size and complexity, the performance could start to degrade, in which case the time may have come to consider the use of **more sophisticated database software which uses a client-server model**. The SQLite web site includes an excellent page on Appropriate Uses For SQLite, including a checklist for choosing between SQLite and client-server databases (https://www.sqlite.org/whentouse.html).

Client-Server databases

Compute Canada offers access to MySQL and Postgres database servers for researchers on both Cedar and Graham (https://docs.alliancecan.ca/wiki/Database_servers):

- Cedar
  - MySQL
    - Description: General purpose server for the researcher wanting to set up SQL tables and issue SQL commands against them.
    - Server name: cedar-mysql-vm.int.cedar.computecanada.ca
    - Short server name: cedar-mysql-vm (can be used instead of long name on most compute nodes)
    - Version: MariaDB version 10.4 Community Edition
    - Documentation: http://www.mariadb.com
  - Postgres
    - Description: General purpose server for the researcher wanting to set up SQL tables and issue SQL commands against them. Includes a PostGIS extension available for those needing to do geocoding.
    - Server name: cedar-pgsql-vm.int.cedar.computecanada.ca
    - Short server name: cedar-pgsql-vm (can be used instead of long name on most compute nodes)
    - Version: PostgreSQL version 10.1, PostGIS version 2.4 extension available
    - Documentation: https://www.postgresql.org and https://postgis.net/documentation
- Graham
  - MySQL
    - Description: General purpose server for the researcher wanting to set up SQL tables and issue SQL commands against them.
    - Server IP Address: 199.241.163.99
    - Server name: cc-gra-dbaas1.sharcnet.ca
    - Version: MariaDB version 10.2
    - Documentation: http://www.mariadb.com

Using modules

Compute Canada servers can execute all software that runs under Linux. In the simplest case, the software you need will already be installed on one of the compute servers. It will then be accessible in the form of a "module". If this is not the case, you can either ask our staff to install it for you, or do it yourself.

Modules are configuration files that contain instructions for modifying your software environment. This modular architecture allows multiple versions of the same application to be installed without conflict.

A "modulefile" contains the information needed to make an application or library available in the user's login session. Typically a module file contains instructions that modify or initialize environment variables such as `PATH` and `LD_LIBRARY_PATH` in order to use different installed programs. Note that the simple fact of loading a module doesn't execute the software in question. To learn the name of the program binary or the syntax for its use, you should read the documentation for this software. By using the module command, you shouldn't normally need to know the exact location or path of the software or library but you can nevertheless see such details about the module by means of the command `module show <module-name>`.

`module`	`spider`	Searches the complete tree of all modules and displays it
	`avail`	List the modules available on a given system
	`list`	Lists the modules that are currently loaded in your environment
	`load`	Lets you load a given module * Example: `module load gcc/9.3 openmpi/4.0`
	`unload`	After the `load` sub-command, `unload` removes a module from your environment **
	`purge`	Remove all the modules you have loaded in one go
	`show`	Displays the entire module
	`help`	Displays a help message
	`whatis`	Shows a description of the module
	`apropros`	Search for a keyword in all modules. If you don't know

	`keyword`	which module is appropriate for your calculation, you can search the description.
	`save`	Save a collection of loaded modules
	`restore`	Restore the saved collection

*If you load a module that is incompatible with one you already have loaded, Lmod will tell you that it has replaced the old module with a new one. This can occur especially for compilers and MPI implementations.

**If you have other modules loaded that depend on this compiler, Lmod will tell you that they have been disabled.

Module collections

Lmod allows you to create a collection of modules. To do so, first `load` the desired modules. For example:

```
module load gcc/9.3 openmpi/4.0.3 mkl
```

Then use the `save` sub-command to save this collection:

```
module save my_modules
```

The `my_modules` argument is a name you give the collection.

Then in a later session or in a job you can restore the collection with the command:

```
module restore my_modules
```

Important note: **We advise against loading modules automatically in your .bashrc**. Instead we recommend that you load modules only when required, for example in your job scripts. To facilitate the repeated loading of a large number of modules we recommend you use a module collection.

When the module system detects two modules of the same family, or two version of the same module, the command module load will automatically replace the original module with the one to be loaded. In the cases where the replaced module is a node in the module hierarchy, dependent modules will be reloaded if there are compatible versions, or deactivated otherwise.

Installing software in your home directory

Most academic software is freely available on the internet. You can email Compute Canada support staff, provide them with a URL, and ask them to install any such package so that you and other

users will be able access it via a module load command. If the license terms and technical requirements are met they will make it available, typically in one or two business days.

Before installing anything ask Marjorie or Valérie for advice.

Running jobs

One of the goals of using a bioinformatics server like Beluga is to launch jobs that require a lot of resources or time. On Compute Canada, there is a job scheduler. You first need to create a job script (details later on). You submit this job script to a piece of software called the scheduler which decides when and where it will run. Once the job has finished you can retrieve the results of the calculation. Normally there is no interaction between you and the program while the job is running, although you can check on its progress if you wish.

The job scheduler is a piece of software with multiple responsibilities. It must:

- maintain a database of jobs,
- enforce policies regarding limits and priorities,
- ensure resources are not overloaded, for example by only assigning each CPU core to one job at a time,
- decide which jobs to run and on which compute nodes,
- launch them on those nodes, and
- clean up after each job finishes.

On Compute Canada clusters, these responsibilities are handled by the Slurm Workload Manager.

On general-purpose (GP) clusters this job reserves 1 core and 256MB of memory for 15 minutes. On Niagara this job reserves the whole node with all its memory. Directives (or "options") in the job script are prefixed with #SBATCH and must precede all executable commands. All available directives are described on the sbatch page. Our policies require that you supply at least a time limit (--time) for each job. You may also need to supply an account name (--account).

Example of job: File = `simple_job.sh`

```
#!/bin/bash

#SBATCH --time=00:15:00

#SBATCH --account=rrg-someuser

echo 'Hello, world!'

sleep 30
```

Submitting a job:

```
sbatch simple_job.sh
```

Listing jobs:

```
sq

squeue -u <username>

squeue -u <username> -t RUNNING

squeue -u <username> -t PENDING
```

The ST column of the output shows the status of each job. The two most common states are "PD" for "pending" or "R" for "running".

Show detailed information for a specific job:

```
scontrol show job -dd <jobid>
```

Cancelling job:

`scancel <jobid>` or `scancel -u $USER` or `scancel -t PENDING -u $USER`

Options for submitting a job to Slurm

`--account=<account>`	Charge resources used by this job to specified account. The account is an arbitrary string. The account name may be changed after job submission using the `scontrol` command.
`--begin=<time>`	Submit the batch script to the Slurm controller immediately, like normal, but tell the controller to defer the allocation of the job until the specified time. Examples: --begin=16:00 --begin=now+1hour --begin=now+60 (seconds by default) --begin=2010-01-20T12:34:00
`--chdir=<directory>`	Set the working directory of the batch script to directory before it is executed. The path can be specified as full path or relative path to the directory where the command is executed.
`--error=<filename_pattern>`	Instruct Slurm to connect the batch script's standard error directly to the file name specified in the "filename pattern". By default both standard output and standard error are directed to the same file.
`--hold`	Specify the job is to be submitted in a held state (priority of zero). A held job can now be released using scontrol to reset its priority (e.g.

	"scontrol release <job_id>").
--mail-type=<type>	Notify user by email when certain event types occur. Valid type values are NONE, BEGIN, END, FAIL, REQUEUE, ALL (equivalent to BEGIN, END, FAIL, INVALID_DEPEND, REQUEUE, and STAGE_OUT)
--mail-user=<user>	User to receive email notification of state changes as defined by --mail-type. The default value is the submitting user.
--mem=<size>[units]	Specify the real memory required per node. Default units are megabytes. Different units can be specified using the suffix [K|M|G|T].
-N, --nodes=<minnodes>[-maxnodes]	Request that a minimum of minnodes nodes be allocated to this job. A maximum node count may also be specified with maxnodes. If only one number is specified, this is used as both the minimum and maximum node count.
-n, --ntasks=<number>	sbatch does not launch tasks, it requests an allocation of resources and submits a batch script. This option advises the Slurm controller that job steps run within the allocation will launch a maximum of number tasks and to provide for sufficient resources. The default is one task per node
--output=<filename_pattern>	Instruct Slurm to connect the batch script's standard output directly to the file name specified in the "filename pattern". By default both standard output and standard error are directed to the same file.
--test-only	Validate the batch script and return an estimate of when a job would be scheduled to run given the current job queue and all the other arguments specifying the job requirements. No job is actually submitted.
--time=<time>	Set a limit on the total run time of the job allocation. A time limit of zero requests that no time limit be imposed. Acceptable time formats include "minutes", "minutes:seconds", "hours:minutes:seconds", "days-hours", "days-

	hours:minutes" and "days-hours:minutes:seconds".

Filename pattern

sbatch allows for a filename pattern to contain one or more replacement symbols.

- \\: Do not process any of the replacement symbols.
- %%: The character "%".
- %A: Job array's master job allocation number.
- %a: Job array ID (index) number.
- %J: jobid.stepid of the running job. (e.g. "128.0")
- %j: jobid of the running job.
- %N: short hostname. This will create a separate IO file per node.
- %n: Node identifier relative to current job (e.g. "0" is the first node of the running job) This will create a separate IO file per node.
- %s: stepid of the running job.
- %t: task identifier (rank) relative to current job. This will create a separate IO file per task.
- %u: User name.
- %x: Job name.

Completed jobs

Get a short summary of the CPU- and memory-efficiency of a job with `seff`:

```
seff 12345678
```

Job ID: 12345678

Cluster: cedar

User/Group: jsmith/jsmith

State: COMPLETED (exit code 0)

Cores: 1

CPU Utilized: 02:48:58

CPU Efficiency: 99.72% of 02:49:26 core-walltime

Job Wall-clock time: 02:49:26

Memory Utilized: 213.85 MB

Memory Efficiency: 0.17% of 125.00 GB

Find more detailed information about a completed job with sacct, and optionally, control what it prints using --format:

```
sacct -j <jobid>

sacct -j <jobid> --format=JobID,JobName,MaxRSS,Elapsed
```

The output from sacct typically includes records labelled .bat+ and .ext+, and possibly .0, .1, .2, ... The batch step (.bat+) is your submission script - for many jobs that's where the main part of the work is done and where the resources are consumed. If you use srun in your submission script, that would create a .0 step that would consume most of the resources. The extern (.ext+) step is basically prologue and epilogue and normally doesn't consume any significant resources.

If a node fails while running a job, the job may be restarted. sacct will normally show you only the record for the last (presumably successful) run. If you wish to see all records related to a given job, add the --duplicates option.

Use the MaxRSS accounting field to determine how much memory a job needed. The value returned will be the largest resident set size for any of the tasks. If you want to know which task and node this occurred on, print the MaxRSSTask and MaxRSSNode fields also.

The sstat command (https://slurm.schedmd.com/sstat.html) works on a running job much the same way that sacct (https://slurm.schedmd.com/sacct.html) works on a completed job.