Solving Optimization Models Using CPLEX Solver on High-Performance Computing (HPC) Systems

Yashar Pourrahimian[1]
[1] School of Mining and Petroleum Engineering, University of Alberta, Edmonton, Canada
Mining Optimization Laboratory (MOL)

ABSTRACT

The high-performance computing (HPC) discipline of computer science utilizes the front line of the computational capacity to run the algorithms faster. The implementation of high-performance computing developed rapidly in many other disciplines besides computer science to solve complex problems. Due to this issue, it is also essential for researchers in the field of Mining engineering to be familiar with this topic and utilize this tool to solve complex problems in mining engineering. This tutorial provides major concepts to implement high-performance computing for mining engineers and those unfamiliar with this area. The first section provides a general introduction to high-performance computing. The second section describes essential concepts such as cluster structure, technical jargon in HPC, and resource scheduling in the area of high-performance computing. The following sections explain resource scheduling and the job submission procedures in detail. This tutorial is useful for solving mathematical optimization problems in mining engineering.

1. Introduction

Computer science approaches are integrated into various engineering practices and applications [12, 16, 22]. In the 21st century, engineering science is passionate about solving more complex engineering problems [3, 6]. Currently, engineering problem-solving is dedicated to designing more complex algorithms to solve complex problems [5, 13, 17, 24]. The complexity of problems and algorithms increases the solution time and provides more challenges to make algorithms or machines faster.

Generally, engineering problems aim to solve optimization or simulation problems. However, assessing the uncertainties and providing the different scenarios are inseparable parts of every engineering problem. Essentially, obtaining reliable results sometimes requires running hundreds of scenarios to achieve reliable and robust results.

Recently High-Performance Computing (HPC) approach implemented in several engineering fields such as social media [25], semantics [9], geology [4], archeology [8], materials[1], urban planning [2], graphics [16], genomics [26], brain imaging [23], economics [19], game design [16], and even music [21]. Moreover, developments in sensing tools provide us with a reliable platform for collecting big data to analyze them. The existence of a considerable amount of data would help us to recognize complex patterns and relations on the underlying problem [2].

During the last decades, mining engineering has constantly been challenged with computational time for solving structurally complex or large-scale problems. These problems are mainly in mine planning by solving the optimization problem [7, 14].Most problems are formulated by linear programming optimization, and the non-linear formulation is converted to the approximate and its linearized equivalent [15, 18, 20]. The problems formulated as Linear (stochastic) mixed-integer

ISBN: 978-1-55195-548-3

optimization models also require high computational time. For instance, solving a large-scale mixed-integer problem (MIP) for an open-pit mine requires hours and even days.

The issues mentioned above reveal the importance of HPC in the engineering area. So, HPC implementation in the mining area would be beneficial to enhance the algorithm and solve computationally expensive problems.

Giving a comprehensive insight into the HPC required an extensive literature review on this area, which is out of the scope of this paper (See [10, 11]). This paper is dedicated to providing a basic tutorial around HPC implementation and, more specifically, parallel computing for the optimization problem.

In the following, the basic concepts of HPC, such as resource scheduling, cluster etc. will be introduced. After that, the procedure for connecting to the cluster, job submission, running a job on the cluster and data transformation will be discussed.

2. Essential Concepts

In this section, the details of the infrastructure and definition of HPC are reviewed, and concepts such as cluster, node, thread, scheduling, job, etc., are explained. The range of definitions has been narrowed as required for our application. Furthermore, as the cluster available for our research is provided by the Digital Research Alliance of Canada, the implementation of the procedure is explained based on their cluster and its policies. The same procedure applies to other clusters with slight changes.

2.1. What is High-Performance Computing?

High-performance computing (HPC) is computing in a "Supercomputer," a computer with high capacity on CPU (or GPU), memory, and storage disks processing millions of operations in a second. The architecture of a supercomputer and its OS system differ from conventional desktop computers.

2.2. What is a Cluster?

To perform the HPC and provide the computational capacity of several CPUs or GPUs, memory and storage disks should connect and perform coherently as a single system. This system is known as a cluster. The fast local or global network usually provides a loose or tightened connection between several clusters. A cluster consists of several nodes (computer servers) that run their shareable operating system.

2.3. What is a Node?

Each node on an HPC system is essentially an individual computer. Each node has a certain amount of sharable memory (typically 64-256 GB per node) and storage disk. Figure 1 illustrates a schematic overview of a cluster.

2.4. What is Job Submission?

Commonly, a user sends several thread(s) to the desktop computer CPU, and then the CPU commences to process the user task(s) and send back the results to the user. This is how a client normally uses the desktop computer for a specific goal, such as running codes, drawings, etc. However, as in HPC, clusters are typically shared between several users simultaneously; we need to send our command to the cluster in an understandable way for the other users and the CPUs. The job submission procedure implements this process in which a script introduces our task(s) to the cluster network(s). The process of sending a task to the cluster to run a specific code or algorithm and receive the results is called "Job submission". The scheduler controls this process (Figure 1).

2.5. What is the Scheduler?

The details of resource scheduling for the cluster are beyond the scope of this paper, but it is essential to be familiar with the workflow of the job schedulers. Usually, clusters are available for several people to take the computational advantage of supercomputers for their own computational approach. It means several users are always eager to run their jobs on the same cluster environment shared by other users.

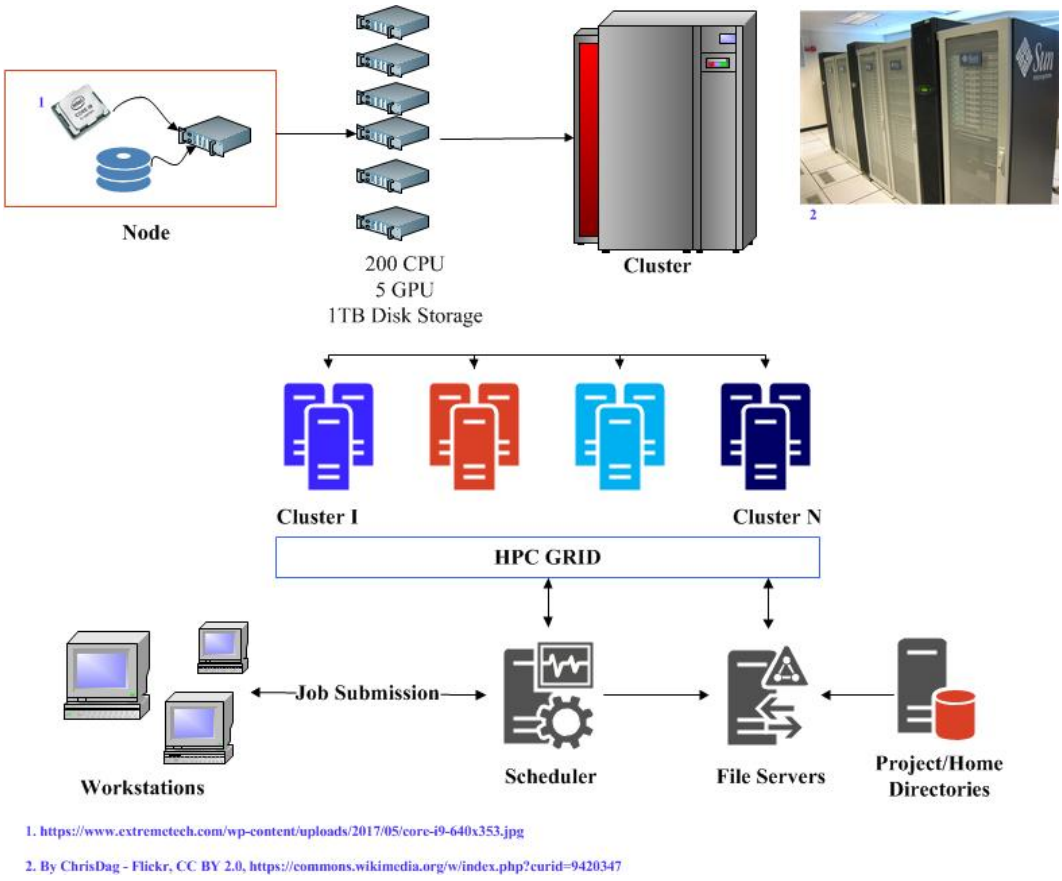

Figure 1. Schematic structure of a cluster.

The scheduler will decide when a job starts based on the user's defined parameters on a job submission batch file. The scheduler always makes a trade-off between available resources and job submission requirements. For example, a job with a longer computational time and lower required resources (number of CPU and amount of memory) may have a higher priority than a job with a shorter computational time and higher resource allocation (Figure 1)

3. Digital Research Alliance of Canada

The Digital Research Alliance of Canada (Alliance) serves Canadian researchers intending to advance Canada's global knowledge economy leader position. By integrating, championing, and funding the infrastructure and activities required for advanced research computing (ARC), research data management (RDM), and research software (RS), they provide the platform for the research community to access tools and services faster than ever before. The Alliance is a nonprofit organization funded by the Canadian government.

ARC is an important component of Canada's digital research infrastructure and is used by the public and private sectors for innovation. The term refers to the elements required to perform computationally and data-intensive research and data management, including HPC and storage. ARC relies on high-speed networks, software, standards, and data-management services.

ARC researchers come from science and industry. They come from all disciplines, including engineering, medical, biological, and life sciences, chemistry and biochemistry, physics, environmental and Earth sciences, humanities and social sciences, and astronomy.

Collaborations between researchers and industry frequently occur in the ARC realm and cut across many sectors, including finance, advanced materials and manufacturing, artificial intelligence, drug development, and aerospace.

3.1. Compute Canada Database (CCDB) Account

Users can access Alliance resources through the Compute Canada Database (CCDB) web portal. The CCDB is the single access source for all Alliance resources.

First, user needs to get an account. The CCDB has 12 classifications of roles, which can be seen on the registration page. All undergraduates, MSc, PhD, PDF, external collaborators, and visiting faculty are considered sponsored users. Sponsored users must be identified and approved by a Principal Investigator (Academic supervisor with an account with CCDB) for access to resources and services. The approval process can take up to 2 business days. When a CCDB registration has been approved, a confirmation email will be sent that includes user's Role Identifier (RI). For registration, use the link below:

https://alliancecan.ca/en/services/advanced-research-computing/account-management/apply-account

After getting an account, the user can use the available resources.

3.2. How to connect a CCDB's server

Secure Shell (SSH) is a widely used standard to connect to remote machines securely. An alternative way to make this connection is to use a software called MobaXterm, which provides all the essential remote network tools (SSH, X11, RDP, VNC, FTP, MOSH, ...) and Unix commands (bash, ls, cat, sed, grep, awk, rsync, ...) for windows users.

Compute Canada's general-purpose clusters are **Béluga**, **Cedar**, **Graham**, and **Narval**. Information about other clusters, login node names, and node characteristics can be found on the Alliance Canada portal or wiki pages (https://docs.alliancecan.ca/wiki/Technical_documentation). Table 1 shows a summary of these clusters.

Table 1. Information about Compute Canada's general-purpose clusters.

Cluster	Description	Location	Information
Béluga	a general-purpose cluster designed for a variety of workload	École de technologie supérieure Montreal	**Availability:** March 2019 **Login node:** beluga.alliancecan.ca **Globus Endpoint:** computecanada#beluga-dtn **Data Transfer Node (rsync, scp, sftp,...):** beluga.alliancecan.ca
Cedar	a heterogeneous cluster suitable for a variety of workloads	Simon Fraser University	**Availability:** Compute RAC2017 allocation started June 30, 2017 **Login node:** cedar.alliancecan.ca **Globus endpoint:** computecanada#cedar-globus **System Status Page:** https://status.alliancecan.ca/
Graham	a heterogeneous cluster, suitable for a variety of workloads	University of Waterloo	**Availability:** In production since June 2017 **Login node:** graham.alliancecan.ca **Globus endpoint:** computecanada#graham-globus **Data transfer node (rsync, scp, sftp,...):** gra-dtn1.alliancecan.ca
Narval	a general-purpose cluster designed for a variety of workloads	École de technologie supérieure Montreal	**Availability:** Since October 2021 **Login node:** narval.alliancecan.ca **Globus Collection:** Compute Canada - Narval **Data transfer node (rsync, scp, sftp,...):** narval.alliancecan.ca
Niagara	a homogeneous cluster intended to enable large parallel jobs of 1040 cores and more	University of Toronto & operated by SciNet	**Availability:** Since April 2018 **Login node:** niagara.alliancecan.ca **Globus Collection:** computecanada#niagara **Data mover nodes (rsync, scp, ...):** nia-dm2, nia-dm2 see moving data **System Status Page:** https://docs.scinet.utoronto.ca

In this paper, the cedar cluster is used to explain the procedure. Cedar has a total of 94,528 CPU cores for computation and 1352 GPU devices. The complete information about the cedar cluster properties can be found on the cedar wiki page: https://docs.alliancecan.ca/wiki/Cedar/en

The primary step to connect a cluster is to know the login node. A login node is a node that functions as a head node or a gateway on the cluster. Users can perform them at manipulation on the login node. The shared file system is also stored on the login node database. Login node for all the compute Canada clusters are following the same pattern as follows (4th column of Table 1):

<clustername>.alliancecan.ca e.g. cedar.alliancecan.ca

1) Download MobaXterm and install it.
2) Execute MobaXterm and after finishing the installation, execute it.
3) In the opened window, choose **Session**, and then select **SSH**, and put the Cedar cluster's log in information (Figure 2).
    a. Remote host: **cedar.alliancecan.ca**
4) The SSH terminal will be opened by clicking the Open button, and the user name and password will be asked for.
    a. Username: is the username that the user has under Compute Canada account, for example

    b. Password: is the same one used to log in to CCDB
    c. The user needs to create an SSH-browser password. This must be provided every time for logging in to the cluster.

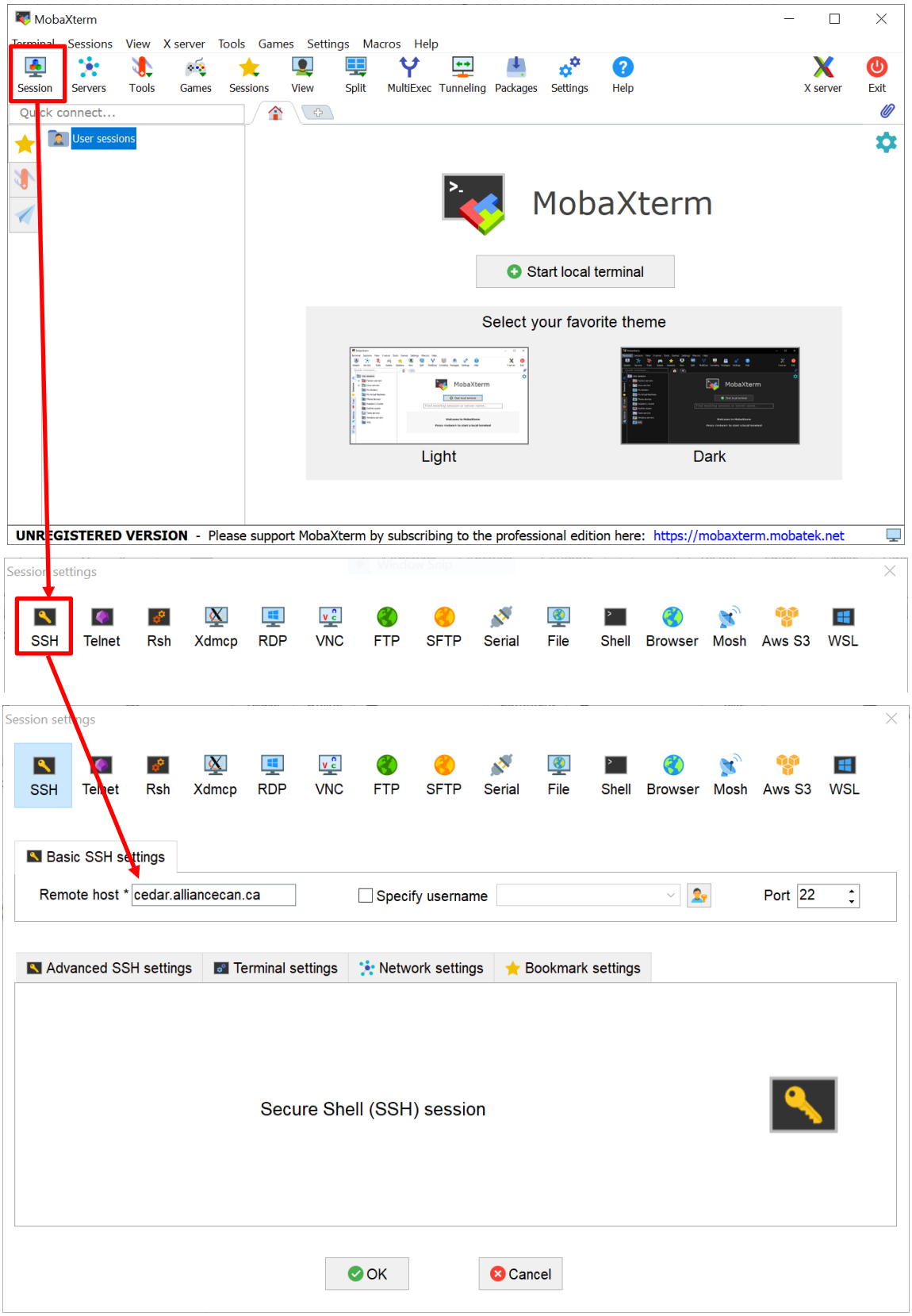

Figure 2. MobaXterm configuration for cedar cluster SSH connection.

After logging in, the user will receive a welcome message indicating that the SSH connection between our desktop computer and the cluster was established successfully (Figure 3).

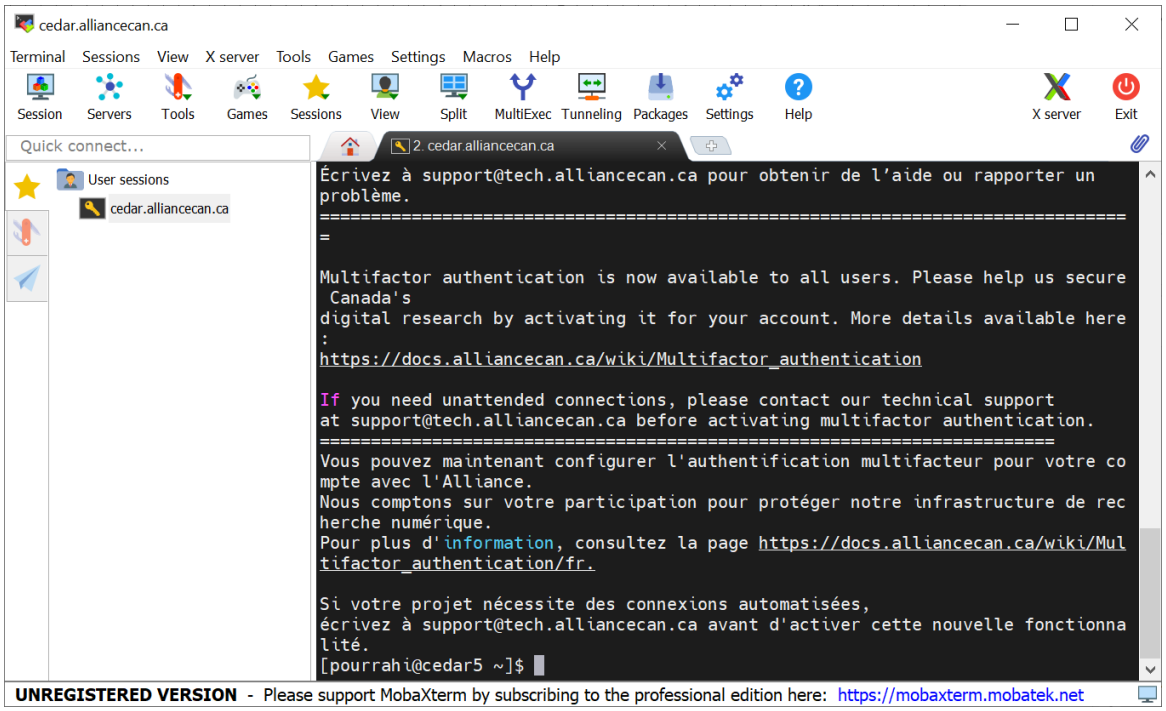

Figure 3. The initial window of Mobaxterm.

To check the connection establishment and see the available logic nodes, type *lscpu* on the command line (Figure 4).

```
[name@server ~]$ lscpu
```

Two types of nodes are available on the node system: login nodes and compute nodes. *Lscpu* prints the information on processors on login nodes (Figure 4). The login node is the launch point to connect to the cluster (gateway), but the compute node is the computational unit of the cluster.

Figure 4 indicates 16 core processors per node and two threads per core. The processor model is Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz.

To obtain the exact amount of available memory type *head -1 /proc/meminfo* on the command line (Figure 5).

```
[name@server ~]$ head -1 /proc/meminfo
```

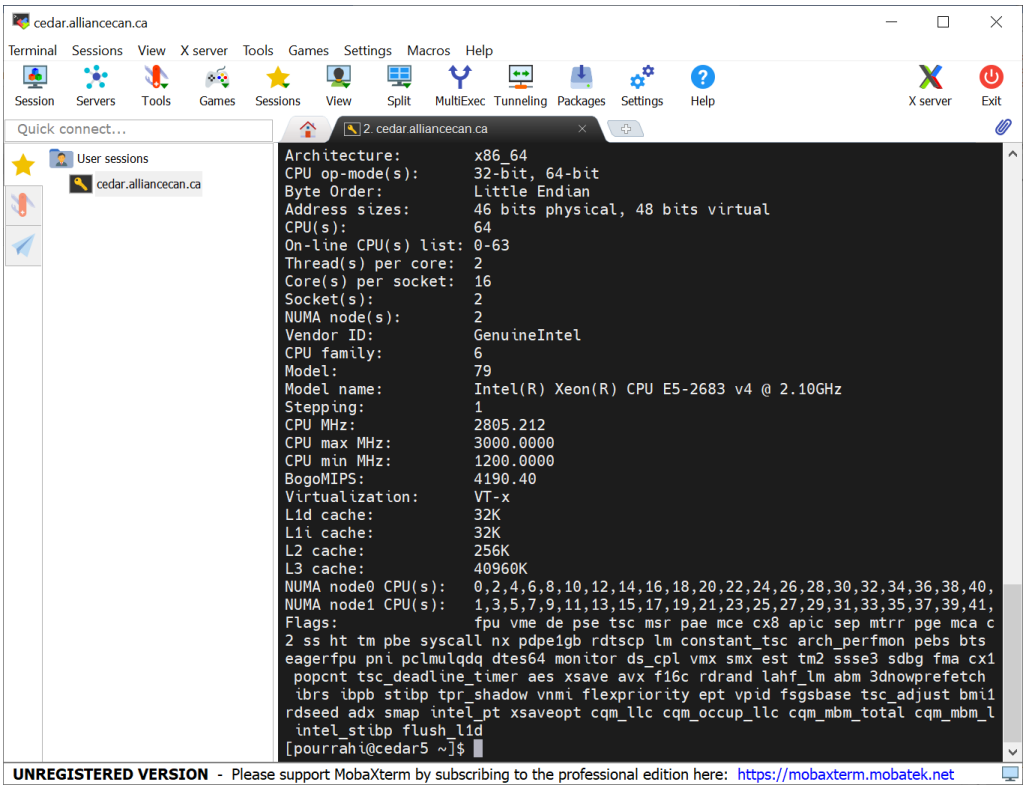

Figure 4. Node information.

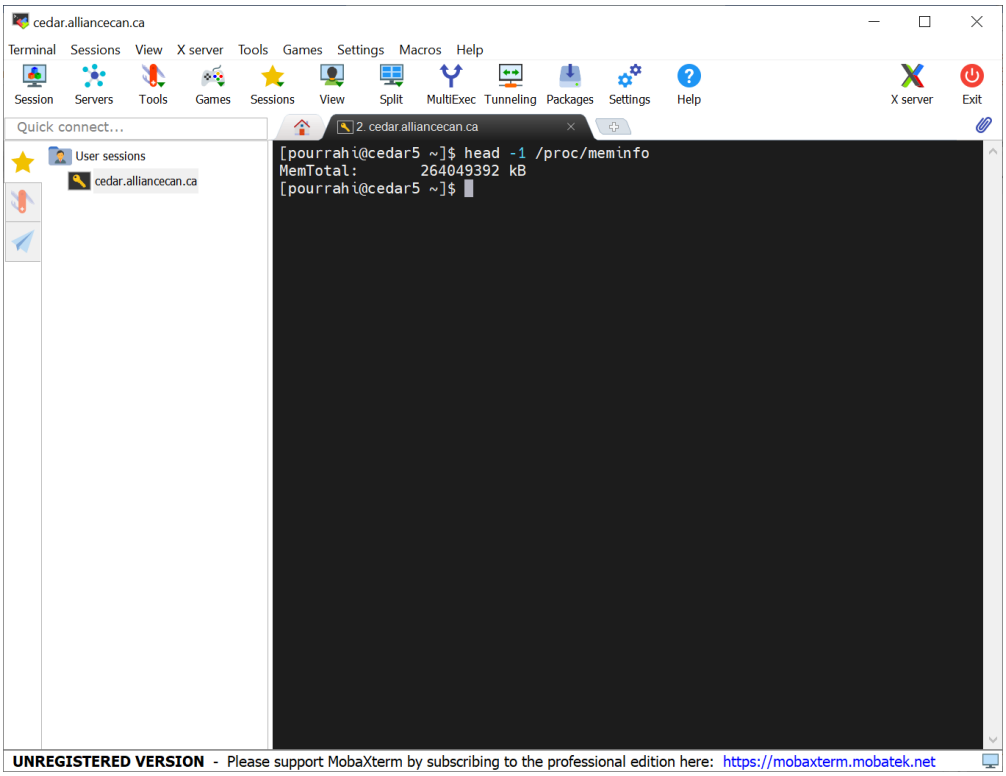

Figure 5. Available memory on the login node.

4. File Transfer

Transferring files is a frequent task for all users to and from an HPC system. This paper will explain the Globus interface to perform file transformation.

4.1. Globus

Globus is a nonprofit service for secure and reliable research data management. Globus provides a transfer API to monitor the progress of a user's file transfer task and manages the endpoint. An endpoint is a remote computing device that communicates back and forth with a network to which it is connected. A complete installation guide for Globus is available at: https://docs.globus.org/how-to/globus-connect-personal-windows/.

1) Follow the installation guide in the link above and install Globus on user's computer.
2) Launch the program at the end of the installation to see the Globus Connect Personal Setup page and then push the login button (Figure 6).

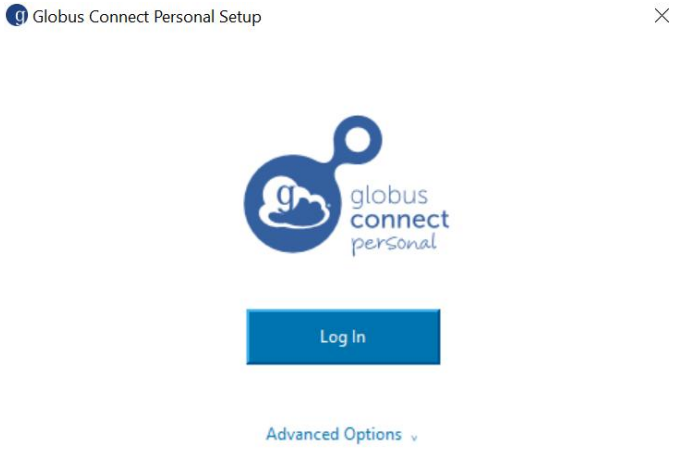

Figure 6. Globus Connect Personal Setup login page.

3) A window will be opened in user's browser. At the top of the window, under **Use your existing organizational login,** select **University of Alberta,** and then press **Continue**.
4) Log in using university CCID and Password, and then press **Yes, continue button**.
5) In the opened window, there is a name in the box in front of "Provide a label for future reference". The user can change or keep it. This name will help the user know from what computer he/she is connecting (Figure 7), then press **Allow**.

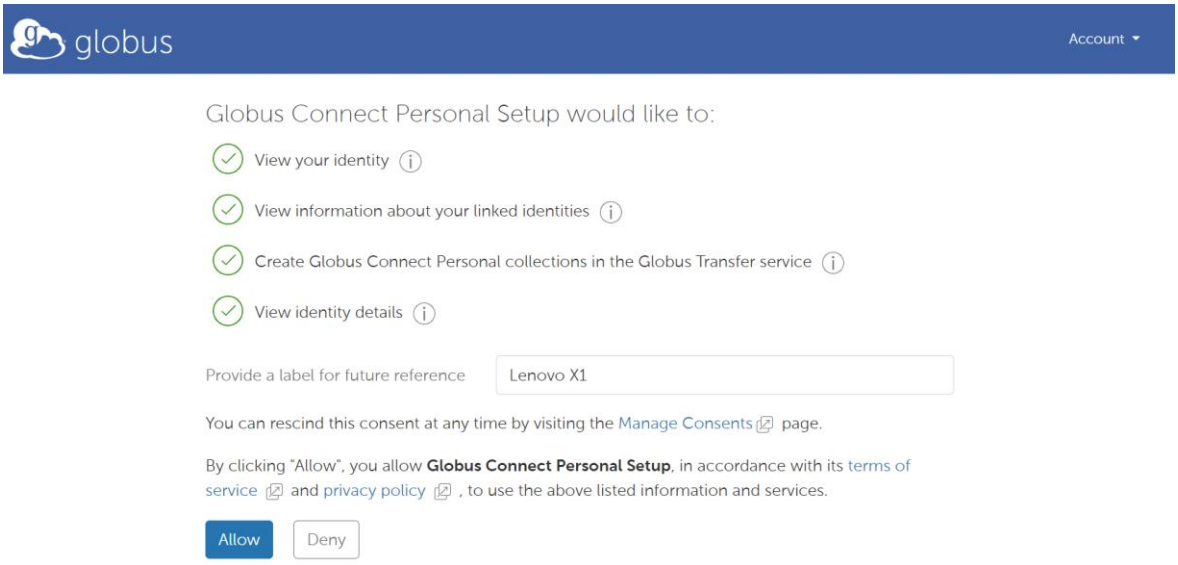

Figure 7. Globus Connect Personal Setup window.

6) In the Collection Details window, give a name for the collection, describe it, and **save** it (Figure 8).

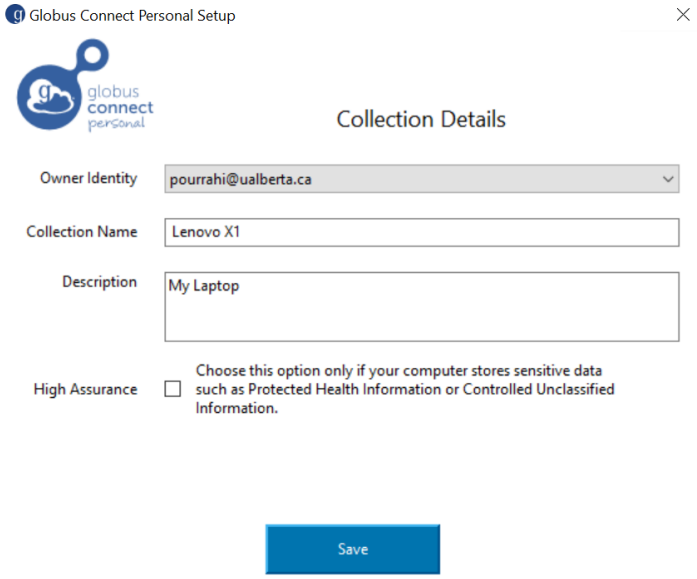

Figure 8. Collection Details window.

7) After completing the setup, a window appears, mentioning that the setup was successful and the Globus Connect Personal is running in the taskbar's notification area. **Press Exit the setup**.

8) Go to the notification area of the taskbar (the bottom right of the monitor under hidden icons) and find the Globus icon (Figure 9).

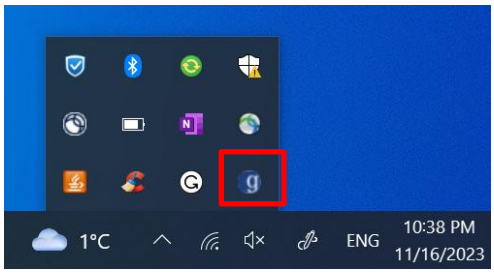

Figure 9. Globus icon at the bottom right of the monitor.

9) Right-click the icon and select "Web: Transfer Files" (Figure 10). A new window will appear in user's web browser (Figure 11). This environment will enable the user to transfer files from local computer to the other endpoints. In the opened window at the top right corner, choose "set two pane" from the panels section.

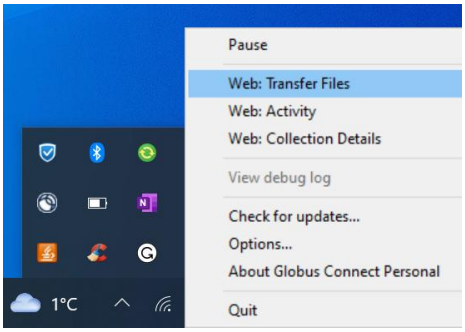

Figure 10. Globus-Web: Transfer Files.

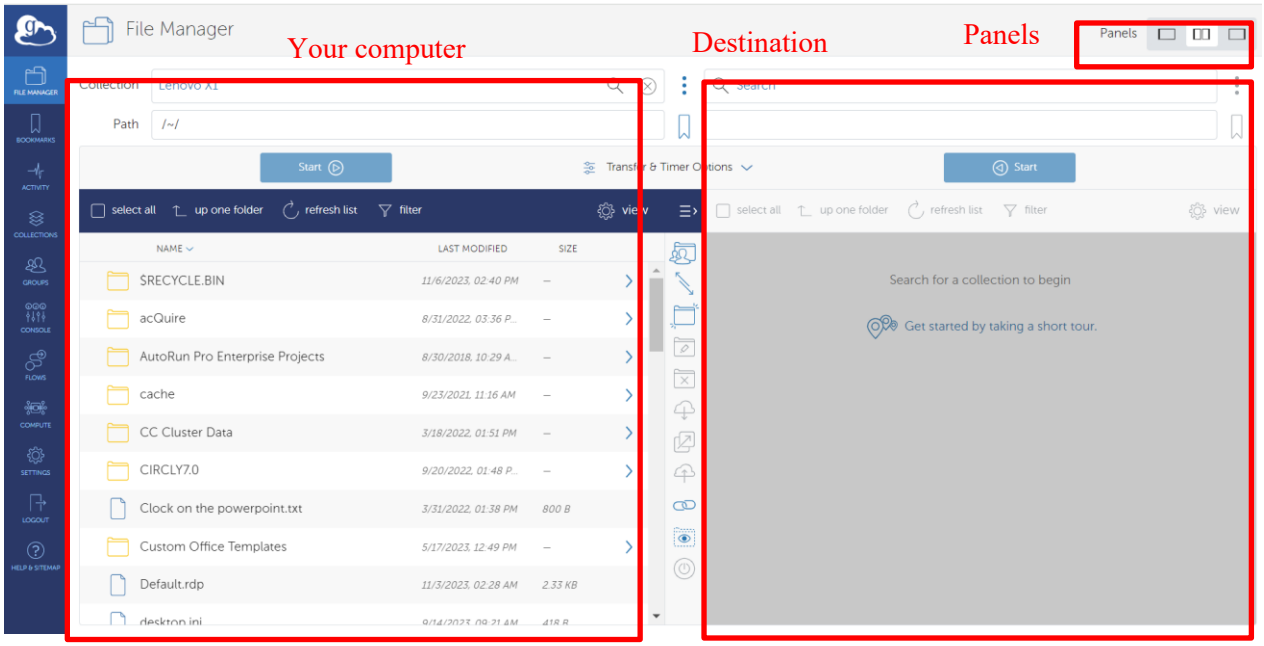

Figure 11. Globus file manager environment.

10) Go to the taskbar's notification area, right-click the Globus Connect Personal icon, and select "Options" to configure Globus Connect Personal (Figure 12).

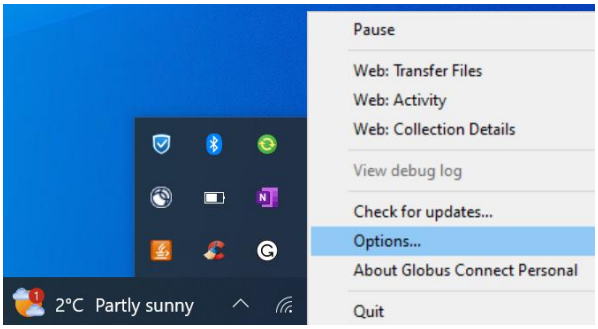

Figure 12. Globus-Options.

11) The options window has five tabs; the most important (and commonly used) are "Access" and "General". The "Access" tab lists folders that will be accessible via Globus for file transfer and sharing. By default, the only folder listed is user's home directory. The user can create a folder and give access to that. Add folders by clicking the "+" icon and selecting the folder he/she wish to make accessible. Then, add it to the accessible list and check the "Sharable" box to share a folder (Figure 13).

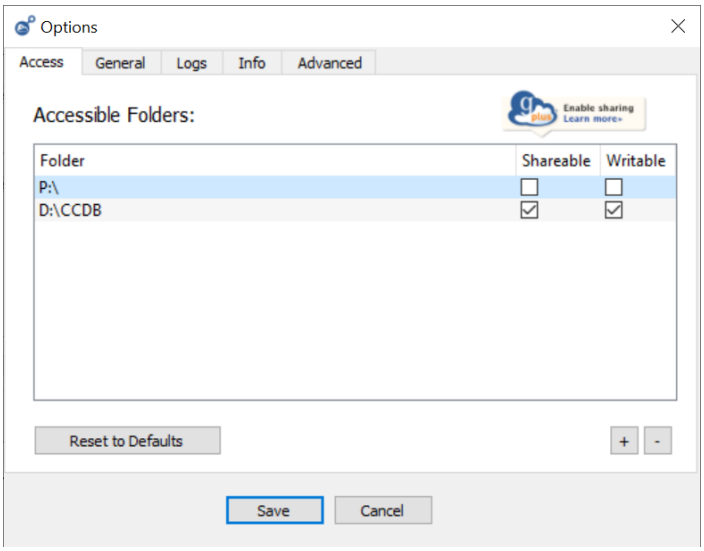

Figure 13. Globus 's "Access" tab under options.

12) The "General" tab allows the user to specify whether he/she wants Globus Connect Personal to run when Windows starts and whether the software should automatically check for updates. It is recommended that the user leaves the "Automatically check for updates" box checked to ensure that the most stable and secure version of Globus Connect Personal is running (Figure 14).

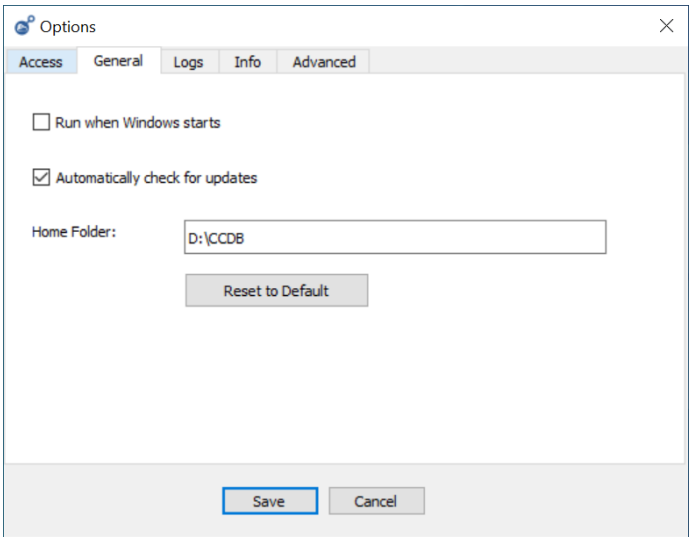

Figure 14. Globus 's "General" tab under options.

13) Now, select the computer as an endpoint on one side and choose the cedar endpoint (computecanada#cedar-globus) on the other side (see Table 1). The endpoint selection should be like Figure 15. By dragging and dropping the files from one side to the other, the files are transferred between the user side and the HPC system (cedar cluster). The login node consists of several folders.

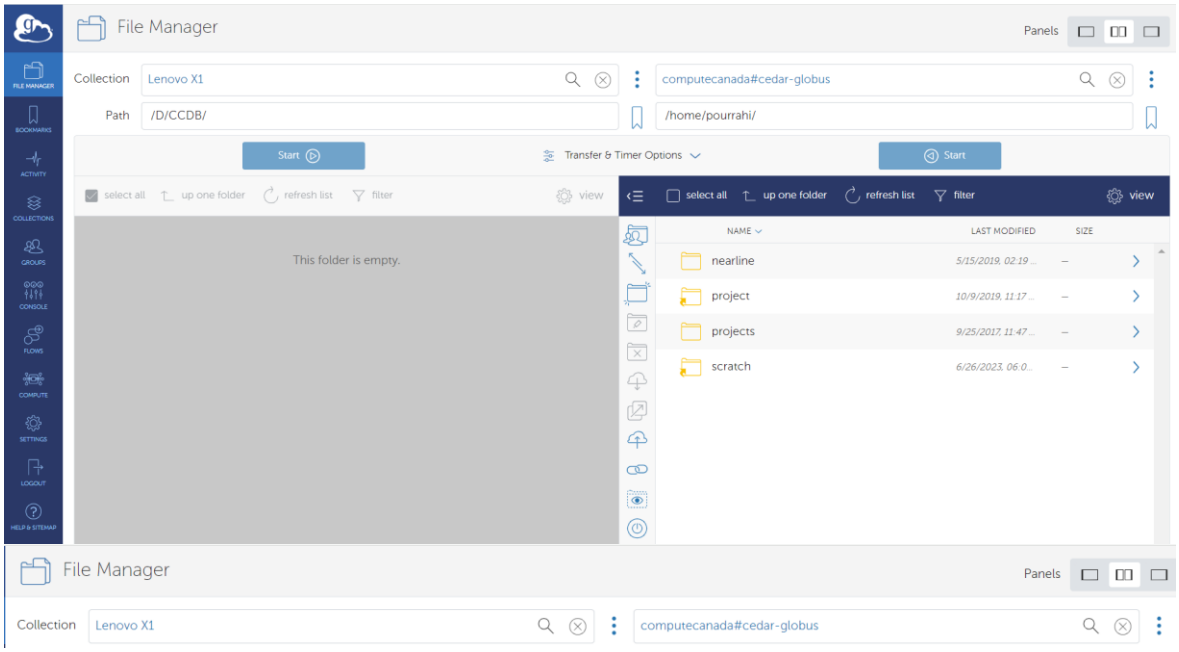

Figure 15. Web browser after endpoint selection.

The **projects** folder is a permanent folder to save user's projects, and the **scratch** directory expires every two months.

5. IBM ILOG CPLEX

CPLEX is a software for optimization developed by IBM and available for academic users. To use CPLEX on Compute Canada clusters, the user must first register with IBM and then download a

personal version of the software. If there is a choice of architectures, choose **Linux x86-64**. Depending on the downloaded version, the name of the file can be like this:

cplex_studio2211.linux_x86_64.bin

After downloading CPLEX, upload the file on the cluster.

1) Create a folder and call it CPLEX, upload the downloaded file into the CPEX folder. The directory is as follows: /home/username/CPLEX/
2) Now, the uploaded binary file should be executable.
    a. In the MobaXterm go to the directory in which the CPLEX installer file is copied.

```
[name@server ~]$ cd CPLEX
```

```
[name@server CPLEX]$
```

    b. Make the binary file executable.

```
[name@server CPLEX]$ chmod +x cplex_studio2211.linux_x86_64.bin
```

    c. Execute the binary file to install CPLEX on the cluster

```
[name@server CPLEX]$ bash ./cplex_studio2211.linux_x86_64.bin
```

    Follow the steps to complete the installation. Install CPLEX in the following directory:

    /home/username/CPLEX/

    d. To use the Python API of CPLEX, such as DOcplex. it is required to load the Python module first. Then, the virtual environment is activated to execute the setup python file. If the CPLEX folder is not the active root, the directory must be changed.

```
[name@server]$ cd CPLEX
```

```
[name@server  CPLEX]$ module load python/3.9
```

```
[name@server  CPLEX]$ virtualenv --no-download ENV
```

```
[name@server  CPLEX]$ source ENV/bin/activate
```

```
(ENV) [name@server  CPLEX]$ python /home/username/CPLEX/python/setup.py install
```

This will also create a new directory, "***ENV***" in the current directory. Finally, execute the following command to install DOcplex.

```
(ENV) [name@server  CPLEX]$ pip install docplex
```

Also, make sure that you upgrade pip and install pandas and pyarrow.

```
(ENV) [name@server  CPLEX]$ pip install --no-index --upgrade pip
```

```
(ENV) [name@server  CPLEX]$ pip install pandas
```

```
(ENV) [name@server  CPLEX]$ pip install pyarrow
```

To exit from the virtual environment, use the following command.

```
(ENV) [name@server  CPLEX]$ deactivate
```

3) To access CPLEX, a personal module should be created. To find our installed CPLEX modules, the configuration file $HOME/.bashrc should be modified to point to the root of this hierarchy.

```
[name@server  CPLEX]$ export MODULEPATH=$HOME/modulefiles:$MODULEPATH
```

4) Now, create a directory structure where the new CPLEX module is located (Figure 16. Created folders on the server).

```
[name@server  CPLEX]$ mkdir -p $HOME/modulefiles/mycplex
```

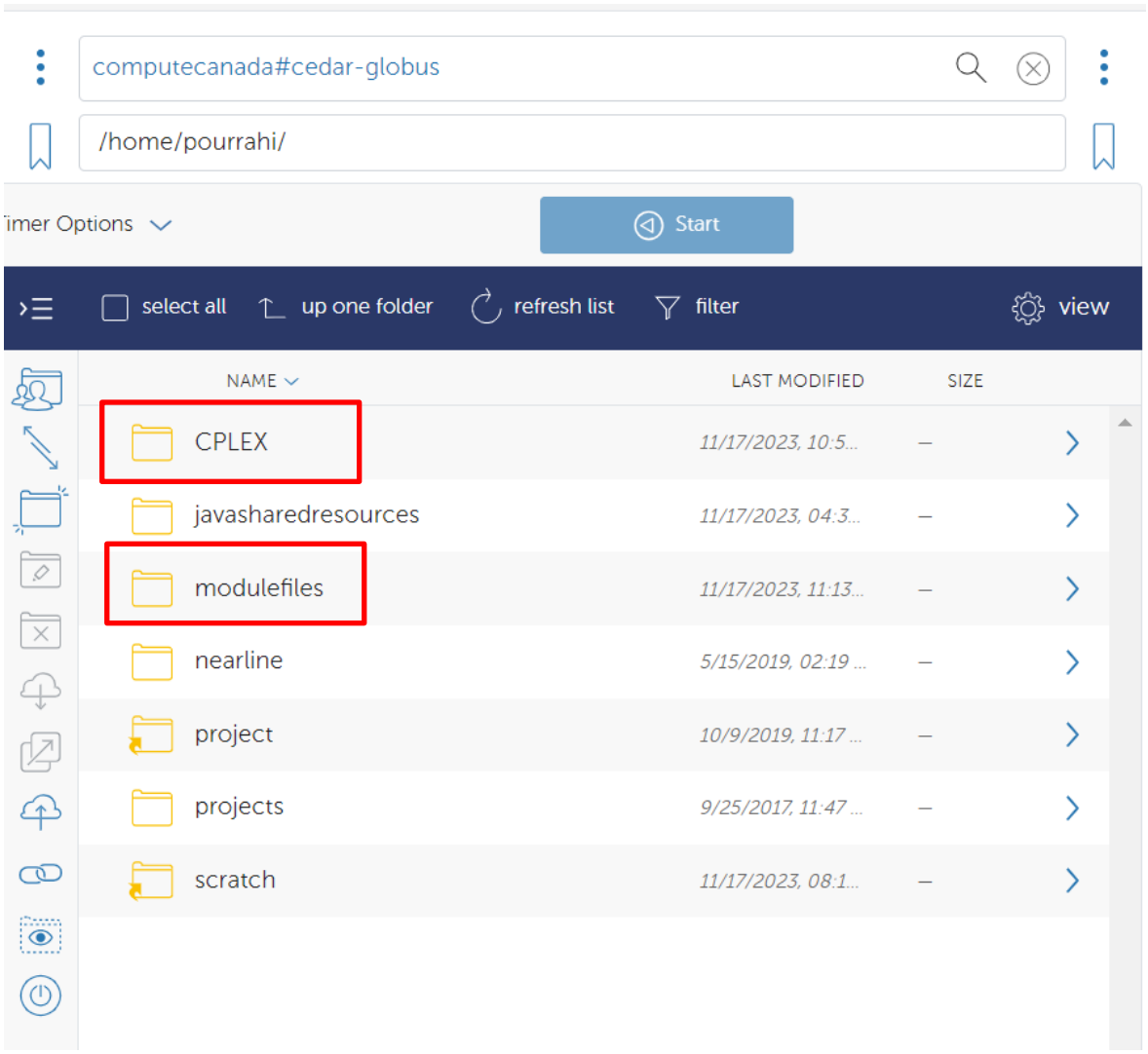

Figure 16. Created folders on the server.

5) Now, go to the "mycplex" directory.

```
[name@server  CPLEX]$ cd ..
```

```
[name@server  ~]$ cd modulefiles
```

```
[name@server  modulefiles]$ cd mycplex
```

```
[name@server  mycplex]$
```

6) Create a directory in the current directory. The name of this directory is the same as the downloaded CPLEX version from IBM (in this paper: **22.1.1**).

```
[name@server  mycplex]$ mkdir 22.1.1
```

A file should be uploaded into the 22.1.1 folder. The file name is "**cplex_module.sh**". Open a text editor copy and paste the following blue text into the file. After creating "**cplex_module.sh**", it should be uploaded into folder 22.1.1 on the server using Globus.

(Path: **/home/username/modulefiles/mycplex/22.1.1/)** The user can do this easily using Globus.

```
#%Module1.0####
##
## cplex
##
proc ModulesHelp { } {
        global cplexversion

puts stderr "\tIBM ILOG cplex "
        puts stderr "\tThis module provides configuration for cplex, concert, cpoptimizer and opl"
}

module-whatis   "IBM ILOG cplex (cplex, concert, cpoptimizer, opl). This version doesn't ask for a
licence file."

# for Tcl script use only
set     cplexversion        22.1.1
set     studio_root         /home/pourrahi/CPLEX/
set     cplexroot           $studio_root/cplex
set     concertroot          $studio_root/concert
set     oplroot            $studio_root/opl
set     cpoptimizerroot      $studio_root/cpoptimizer

set cplexbin x86-64_linux
set cplexlib $cplexbin/static_pic
set concertbin x86-64_linux
set concertlib $concertbin/static_pic
set oplbin x86-64_linux
set opllib $oplbin/static_pic
set cpoptimizerbin x86-64_linux
set cpoptimizerlib $cpoptimizerbin/static_pic

prepend-path    PATH        $cplexroot/bin/$cplexbin
prepend-path    PATH        $oplroot/bin/$oplbin
prepend-path    PATH        $cpoptimizerroot/bin/$cpoptimizerbin

prepend-path    CPATH       $cplexroot/include
prepend-path    CPATH       $concertroot/include
prepend-path    CPATH       $oplroot/include
prepend-path    CPATH       $cpoptimizerroot/include

prepend-path -d " "   CPATH_EXPANDED       -I$cplexroot/include
prepend-path -d " "  CPATH_EXPANDED       -I$concertroot/include
prepend-path -d " "   CPATH_EXPANDED       -I$oplroot/include
prepend-path -d " "   CPATH_EXPANDED       -I$cpoptimizerroot/include

prepend-path    LIBRARY_PATH $cplexroot/lib/$cplexlib
```

```
prepend-path    LIBRARY_PATH $concertroot/lib/$concertlib
prepend-path    LIBRARY_PATH $oplroot/lib/$opllib
prepend-path    LIBRARY_PATH $oplroot/bin/x86-64_linux/
prepend-path    LIBRARY_PATH $cpoptimizerroot/lib/$cpoptimizerlib

prepend-path -d " "   LIBRARY_PATH_EXPANDED -L$cplexroot/lib/$cplexlib
prepend-path -d " "  LIBRARY_PATH_EXPANDED -L$concertroot/lib/$concertlib
prepend-path -d " "   LIBRARY_PATH_EXPANDED -L$oplroot/lib/$opllib
prepend-path -d " "   LIBRARY_PATH_EXPANDED -L$oplroot/bin/x86-64_linux/
prepend-path -d " "   LIBRARY_PATH_EXPANDED  -L$cpoptimizerroot/lib/$cpoptimizerlib

prepend-path    LD_LIBRARY_PATH      $cplexroot/bin/$cplexbin
prepend-path    LD_LIBRARY_PATH      $oplroot/bin/$oplbin

prepend-path     CLASSPATH $cplexroot/lib/cplex.jar
prepend-path     MATLABPATH $cplexroot/matlab
prepend-path     STUDIO_ROOT $studio_root
```

Note: depending on the installed CPLEX version, the red lines correspond to the variables "cplexversion″ and "studio_root″ must be updated.

6. Scheduling jobs

Running tasks (jobs) on a cluster differs greatly from running files on a local computer. The users use a logic node to install the packages, python environments, and files, but the jobs will run on compute nodes. The basic form of job submission is non-interactively through the command line. Any command running on a cluster is a job, and running jobs using the Scheduler is called "batch job submission".

To run a task on a personal computer using Python, the following syntax is used:

```
[user@laptop ~]$ python pythonfile.py
```

This syntax runs the file on a local computer, while the procedure is totally different from running a pythonfile.py on a cluster.

To run a job (file) on a cluster, the user should provide a batch file and submit the job to the Scheduler. After submitting the job, the Scheduler will take the job, put it in a queue, and wait until proper resources are available to run the job based on user definition. Compute Canada clusters use SLURM as a Scheduler.

An example of a batch file for running a mathematical optimization model is presented below. Open Notepad++, then copy and paste the text inside the box into Notepad++.

```
#!/bin/bash
#SBATCH --account def-pourrahi
#SBATCH --mem=30G
#SBATCH --nodes=1
#SBATCH --cpus-per-task=40
#SBATCH --job-name myTest_optimization
#SBATCH --time=23:59:00
#SBATCH --output=myTest_output_%j.txt
#SBATCH --mail-user=pourrahi@ualberta.ca
#SBATCH --mail-type=ALL


#Second part of the batch file.
        source /home/pourrahi/CPLEX/ENV/bin/activate
        module load mycplex/22.1.1

#Run the optimization code
        python ./RunOptimizer.py

#Deactivate the Python virtual environment on the logic node
        deactivate
```

Line 1 tells LINUX that this file is a bash file. This line should be placed in the first line of every bash script file. #SBATCH submits the command to Slurm; notice that # sign before SBATCH is part of the syntax, which is different from a comment on the line on LINUX.

Line 2 specifies the account that will be charged due to resource allocation. This account is always the main account on the cluster account, and all the sub-accounts under the main account supervision use the main account resources.

Line 3 requests the required memory per node; the default value for memory is gigabytes, and 15G means 15 gigabytes per node. It is essential to specify memory correctly.

- If the user does not ask for enough, and the job needs more, the job will be killed.
- If the user asks for too much, it will take a much longer time to schedule a job, and the resources will be wasted.
- If the user asks for more memory than is available on the cluster, the job will never run. The scheduling system will not stop the user from submitting such a job or even send any warning.

Always ask for slightly less than the total memory on a node, as some memory is used for the OS, and the job will not start until enough memory is available.

Line 4 specifies the number of nodes required to perform the job; it can be specified with a min and max values or just a number which will be considered as a min and max number of nodes.

Line 5 sets the number of CPU that is required for each task; the default value is one cpu per each task. Here 40 CPUs have been requested.

Line 6 gives a name for the job.

Line 7 specifies the run time for the cluster scheduler. Specifying the exact time is a trial and error process, an estimation for the first run, and can be optimized after several runs. Longer run-time causes a longer queue time on a scheduler queue. Each cluster has its own time limit. These limits have been summarized in Table 2.

Table 2. The time limit for the CCDB's clusters.

Cluster	Béluga	Cedar	Graham	Narval	Niagara
Maximum run time (day)	7	28	7	7	1

Line 8 ensures that the output log file in **myTest_output_%j.txt** where %j would be job ID assigned by the Scheduler.

Line 9 specifies the email for the user who wants to receive the notifications. The type of notification is defined after defining the target email in the following line.

Line 10 can be one of the following options or a few in several lines. Considering the used code in this example, the server will notify whenever a job is BEGIN, END, FAIL, or REQUEUE.

```
#SBATCH – mail-type=BEGIN
#SBATCH – mail-type=END
#SBATCH – mail-type=FAIL
#SBATCH – mail-type=REQUEUE
#SBATCH – mail-type=ALL
```

After setting up the requirement to run a model on the cluster in the second part of the batch file, the created ENV is activated, and the created module is loaded. Then, another file (**RunOptimizer.py**) in which there is the codes for running the optimization model is executed, and finally, after finishing the optimization, the activated ENV is deactivated.

Before saving the file, go to **Edit > EOL Conversion** and select **Unix (LF)**. By doing this, when the file is saved, it will be saved with UNIX-style line endings. Save the file as a Unix script file and call it **iniRun.sh**.

RunOptimizer.py is the code in which the required libraries are imported, and the created **model.lp** in Python is loaded. Then, the required maximum relative mip_gap tolerance is set up, and the model is executed. Finally, the output, which is a vector of the decision variables (solution), is saved in a .csv file. The following text in the box, shows **RunOptimizer.py** file. Transfer the following files into the scratch folder on the server using the Globus.

1) iniRun.sh
2) RunOptimizer.py
3) model.lp

```
import cplex
import pandas as pd
import sys

# Initialize CPLEX
cpx = cplex.Cplex()

# Try reading the model, handle exceptions if the file is not found or other IO errors
try:
    cpx.read("model.lp")  # Update path if necessary
except Exception as e:
    print(f"Error reading model.lp: {e}")
    sys.exit(1)

# Define the maximum allowable relative MIP gap tolerance (e.g., 5%)
max_relative_mip_gap_tolerance = 0.05

class FileLogger:
    def __init__(self, file_object):
        self.log = file_object

    def write(self, message):
        self.log.write(message)

    def flush(self):
        self.log.flush()

# Attempt to open the log file, handle exceptions
try:
    with open("cplex_log.txt", "w") as log_file:  # Update path if necessary
        file_logger = FileLogger(log_file)
        cpx.set_log_stream(file_logger)
        cpx.set_results_stream(file_logger)
        cpx.parameters.mip.tolerances.mipgap.set(max_relative_mip_gap_tolerance)

        relative_mip_gap = 1.0
        while relative_mip_gap > max_relative_mip_gap_tolerance:
            cpx.solve()
            relative_mip_gap = cpx.solution.MIP.get_mip_relative_gap()

                    if relative_mip_gap <= max_relative_mip_gap_tolerance:
            # Store the results in a DataFrame and save it as a CSV file
            df = pd.DataFrame({
                "Variable_names": cpx.variables.get_names(),
                "Variable_values": cpx.solution.get_values()
            })
            df.to_csv("solution.csv")  # Save the DataFrame in CSV format
        else:
            print("No optimal solution found within the specified tolerance.")

except IOError as e:
    print(f"IO Error: {e}")
```

To run a batch file named **iniRun.sh** on a compute node, go to the **/scratch/username**.

```
[name@server]$ cd /scratch/pourrahi
```

Then set the execute permission on the file, allowing it to be run as a program. For this purpose, use the following commands:

```
[name@server pourrahi]$ chmod +x iniRun.sh
```

Then execute the file using the following command:

```
[name@server]$ sbatch iniRun.sh
```

Upon submitting a job, a job ID will be assigned to it, for example, 19074616.

```
Submitted batch job 19074616
```

The general command for checking the status of Slurm jobs is **squeue**, but by default, it supplies the user's own jobs or use the following code (Figure 17):

```
[name@server]$ sq
```

OR

```
[name@server]$ squeue -u <username>
```

```
[pourrahi@cedar5 ~]$ sq
             JOBID     USER      ACCOUNT           NAME  ST  TIME_LEFT NODES CPUS TRES_PER_N MIN_MEM NODELIST (REASON)
          19159867 pourrahi def-pourrahi myTest_optimiz   R 6-17:45:55     1   20        N/A     30G cdr1744 (None)

[pourrahi@cedar5 /]$ squeue -u pourrahi
             JOBID     USER      ACCOUNT           NAME  ST  TIME_LEFT NODES CPUS TRES_PER_N MIN_MEM NODELIST (REASON)
          19159867 pourrahi def-pourrahi myTest_optimiz   R 6-17:33:11     1   20        N/A     30G cdr1744 (None)
```

Figure 17: Checking the status of the submitted job(s).

To show only running jobs, or only pending jobs, Use the following commands:

```
[name@server]$ squeue -u <username> -t RUNNING
[name@server]$ squeue -u <username> -t PENDING
```

To estimate when a certain job is going to start, use the following command:

```
[name@server]$ squeue --start -j <jobID>
```

To check the list of the completed jobs, use the following command:

```
[name@server]$ sacct
```

To cancel a running or submitted job on the cluster, use the following command:

```
[name@server]$ scancel <jobID>
```

To show detailed information for a specific job, use the following command:

```
[name@server]$ scontrol show job -dd <jobID>
```

To get a summary of the CPU and memory efficiency of a job, use the following command:

```
[name@server]$ seff <jobID>
```

7. References

[1] Alowayyed, S., Piontek, T., Suter, J. L., Hoenen, O., Groen, D., Luk, O., Bosak, B., Kopta, P., Kurowski, K., & Perks, O. (2019). *Patterns for high performance multiscale computing.* Future Generation Computer Systems, **91**, pp. 335-346.

[2] Aqib, M., Mehmood, R., Alzahrani, A., Katib, I., Albeshri, A., & Altowaijri, S. M. (2019). *Rapid transit systems: Smarter urban planning using big data, in-memory computing, deep learning, and GPUs.* Sustainability, **11**(10), p. 2736.

[3] Bazilevs, Y., Takizawa, K., & Tezduyar, T. E. (2019). *Computational analysis methods for complex unsteady flow problems.* Mathematical Models and Methods in Applied Sciences, ***29***(5), pp. 825-838.

[4] Chen, J., Hideyuki, O., Takeyama, T., Oishi, S., & Hori, M. (2019). *Toward a numerical-simulation-based liquefaction hazard assessment for urban regions using high-performance computing.* Engineering Geology, **258**, p. 105153.

[5] Chugh, T., Sindhya, K., Hakanen, J., & Miettinen, K. (2019). *A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms.* Soft Computing, **23**(9), pp. 3137-3166.

[6] Dhiman, G. & Kumar, V. (2019). *Spotted hyena optimizer for solving complex and non-linear constrained engineering problems,* in *Harmony Search and Nature Inspired Optimization Algorithms,* Springer, pp. 857-867.

[7] Dimitrakopoulos, R. (2011). *Stochastic optimization for strategic mine planning: a decade of developments.* Journal of Mining Science, **47**(2), pp. 138-150.

[8] Dodd, J. A. (2019). *The application of high performance computing in rock art documentation and research.* Adoranten,

[9] Durbha, S. S., Kurte, K. R., & Bhangale, U. (2017). *Semantics and high performance computing driven approaches for enhanced exploitation of earth observation (EO) data: State of the art.* Proceedings of the National Academy of Sciences, India Section A: Physical Sciences, **87**(4), pp. 519-539.

[10] Eijkhout, V. (2013). *Introduction to High Performance Scientific Computing.* Lulu.com.

[11] García-Risueño, P. & Ibáñez, P. E. (2012). *A review of high performance computing foundations for scientists.* International Journal of Modern Physics C, **23**(7), p.1230001.

[12] Gruenhagen, J. H. & Parker, R. (2020). *Factors driving or impeding the diffusion and adoption of innovation in mining: A systematic review of the literature.* Resources Policy, **65**, p. 101540.

[13] Jankauskas, K., Papageorgiou, L. G., & Farid, S. S. (2019). *Fast genetic algorithm approaches to solving discrete-time mixed integer linear programming problems of capacity*

planning and scheduling of biopharmaceutical manufacture. Computers & Chemical Engineering, **121**, pp. 212-223.

[14] Liu, S. Q. & Kozan, E. (2016). *New graph-based algorithms to efficiently solve large scale open pit mining optimisation problems.* Expert Systems with Applications, **43**, pp. 59-65.

[15] Moreno, E., Ferreira, F., Goycoolea, M., Espinoza, D., Newman, A., & Rezakhah, M. (2015). *Linear programming approximations for modeling instant¬mixing stockpiles*, in *37th International Symposium on Application of Computers and Operations Research in the Mineral Industry,* APCOM 2015. Fairbanks, USA.

[16] Peng, C. (2019). *High-performance computer graphics technologies in engineering applications.* World Journal of Engineering, **16**(2), pp. 304-308.

[17] Ramos, A. R., de Lázaro, J. M. B., Prieto-Moreno, A., da Silva Neto, A. J., & Llanes-Santiago, O. (2019). *An approach to robust fault diagnosis in mechanical systems using computational intelligence.* Journal of Intelligent Manufacturing, **30**(4), pp. 1601-1615.

[18] Rezakhah, M., Moreno, E., & Newman, A. (2019). *Practical performance of an open pit mine scheduling model considering blending and stockpiling.* Computers & Operations Research, **115**, p. 104638.

[19] Schryen, G., Kliewer, N., & Fink, A. (2020). *High performance business computing.* Business & Information Systems Engineering, **62**, pp. 1-3.

[20] Sepúlveda Escobedo, E. M. (2018). *Quantification of uncertainty of geometallurgical variables for mine planning optimisation.* Thesis, University of Adelaide.

[21] Shi, H., Jiang, Z., Liu, Q., and Cai, X. (2019). *An efficient FPGA parallel implementation for 2-D MUSIC algorithm*. in *Proceedings of IOP Conference Series: Earth and Environmental Science*, IOP Publishing, pp. 032168.

[22] Topol, E. J. (2019). *High-performance medicine: the convergence of human and artificial intelligence.* Nature medicine, **25**(1), pp. 44-56.

[23] Xanthis, C. G. & Aletras, A. H. (2019). *coreMRI: A high-performance, publicly available MR simulation platform on the cloud*. PloS one, **14**(5), p. e0216594.

[24] Zhang, X., Ding, F., Xu, L., & Yang, E. (2019). *Highly computationally efficient state filter based on the delta operator.* International Journal of Adaptive Control and Signal Processing, **33**(6), pp. 875-889.

[25] Zheng, S., Shae, Z.-Y., Zhang, X., Jamjoom, H., & Fong, L. (2011). *Analysis and modeling of social influence in high performance computing workloads*. in *Proceedings of European Conference on Parallel Processing*, Springer, pp. 193-204.

[26] Zou, J., Huss, M., Abid, A., Mohammadi, P., Torkamani, A., & Telenti, A. (2019). *A primer on deep learning in genomics.* Nature genetics, **51**(1), pp. 12-18.