Guideline on using SLURM HPC clusters

Abstract

High Performance Computing (HPC) clusters are a crucial tool in any form of academic research which requires strong computational analysis of any set of data. An increasing number of universities now have adopted their own HPC clusters each with their own specific parameters, but generally all being capable of performing high order computational analysis. HPC clusters are comprised of many computational nodes fast-connected to one another, and the activity of these nodes with respect to one another is regulated by workload managers. Each workload manager has its own specifications when it comes to using it and one can not be ranked as more efficient than the other. however they all have the same overall function which is optimally allocating memory and processors to the users jobs. This paper elaborates on the use of SLURM workload manager and the potential use of os.system().

1 Overall architecture of SLURM

SLURM allocates a specific amount of memory to each user, and if a job submitted by a user exceeds the memory threshold, then SLURM cancels it. Another factor which determines the scheduling of a job is also user priority, where users of a higher priority are granted access to more memory. Each jobs is constituted of one or more steps to be run and each job as a whole requires a specific amount of resources. Based on how the steps are related to one another memory-wise and based on the overall amount of memory required, a job is subdivided into tasks with each being allocated a specific amount of resources. The HPC Linux clusters all have a general linux command line but specific knowledge is required when managing jobs, creating environments with specific packages and using specific software as needed within the cluster. The following sections will detail how to install desired software in the cluster, how to create/edit/move files, different methods of submitting jobs on the cluster and creating a virtual Jupyter Notebook environment in the cluster.

1.1 Data tree structure, creating/arranging/editing files

The cluster in itself is a big collection of different types data which can be edited or analysed by the computational nodes. The data in the cluster is arranged on a tree format with a base "root" containing the workload manager configuration which assigns RAM and other memory to each node as needed, a "trunk" containing all the base software which will be shared to the users (python, matlab, ipopt, conda etc) and the "branches" are the memory allocated to each user. Only the cluster managerial staff have access the "root" and "trunk" levels of the cluster. In the data storage perspective, one could think of the cluster as a big folder containing sub-folders in it, which also contain sub-folders and so on. A data path in

the cluster is the same concept as a data path in a computer or laptop. Let's take a situation where a person downloads a file named "some-file" from a website to their device. That file now can be found in "Downloads" and say we want to move that file in "Desktop" and we want to move it in a specific folder which we name "2-folder" located within another folder named"1-folder". We grab the file and move to that folder, so now it can be said that the data-path of that file is "/Desktop/1-folder/2-folder/some-file". In the same way, a HPC clusters operating with SLURM have a data path of the format "/data/users/username" where "data" and "users" represent the data levels accessible only by the clusters managerial staff whereas "username" and whatever comes afterward is accessible by the user. As mentioned, each user is allocated a specific memory space in the cluster and that can be used to store any type of file/data which could be later analyzed by a software in the cluster. This memory space allocated to the user corresponds to the "username" level of the data tree which can be thought of again as a big folder. Same as when a person buys a brand new device, this folder is originally empty and the memory there is at users disposal. Within this directory there are a number of basic actions that can be performed to either a file or a directory: creating one, moving it, copying it, deleting it.

2 SLURM-HPC Commands

2.1 Logging in to HPC account

In almost all cases when a cluster account is being used for academic purposes, a VPN connection must be set up before the user can log in. In order to log in to a cluster, the "ssh" (Secure Shell Protocol) command is used on the following manner:

```
$ ssh username@hpc-logon.nyit.edu
```

. A dialog box should be generated once this command runs which asks for the user password. Please note that when a user types in their password, the characters will not appear on the screen in any form due to safety purposes.

2.1.1 Creating a directory, file

In order to create a directory, use the command

```
$ mkdir adirectory
```

. In order to navigate from one directory to one another, run $ cd "data path of directory". Example: in order to create another directory within "adirectory", we run:

```
$ cd /data/users/username/adirectory
```

```
$ mkdir bdirectory
```

In order to navigate from "bdirectory" to "adirectory" we run:

```
$ cd /data/users/username/adirectory
```

, whereas to move from "adirectory" to "bdirectory", we run only:

```
$ cd bdirectory
```

since "bdirectory" is a sub-directory of "adirectory". In order to create a file we use the command $ touch.
Example:

```
$ touch newfile.txt
```

creates a file called "newfile.txt". In the same way one can create different formats of files by changing the file extension (.py, .edf,.dat etc).

2.1.2 Moving files,directories

In order to move a directory from one location to another, the "mv" command should be used. Example: ADirectory has data path - /data/users/username/Adirectory. Bdirectory has data path /data/users/username/BDirectory. In order to move BDirectory into ADirectory:

```
$ mv /data/users/username/BDirectory /data/users/username/Adirectory
```

or

```
$ mv BDirectory ADirectory
```

given that ADirectory and BDirectory are originally in /data/users/username. The exact same logic applies to moving files from one directory to the other.

2.1.3 Copying files, directory

Copying takes place via the use of the command "cp" and it is a function of many uses. Firstly, one could create a copy of a file and rename it using "cp". Example: a_textfile.txt is located in /data/users/username/ and in order to make a copy of it and name it b_textfile.txt, we run:

```
$ cp a_textfile.txt b_textfile.txt
```

If it is needed to make a copy of a file and move it to another location, "cp" comes in handy since it does not requir ethe use of "mv". Example:

```
$ cp /data/users/username/a_textfile.txt /data/users/username/a_directory/
```

copies a_texfile.txt and moves it to "a_directory". Note: the file will keep the same name unless specified differently at the previous command. The same logic applies when moving one directory from one location to the other, however when copying directories the "-r" option should be used after "cp".
Example:

```
$ cp -r /data/users/username/some_directory /data/users/username/a_directory/
```

copies "somedirectory" and moves it to "a_directory". The "scp" command is a derivation of "cp" and allows for the copying of files and directories from the cluster to a device and vice versa, or from one cluster account to the other. This command has the same principle as "cp" in that it requires an initial location and a destination to be provided as arguments. Example:

```
$ scp -r username@hpc-logon.nyit.edu:/data/users/username/a_file.txt /Desktop
```

takes a file named "a_file.txt" located in "/data/users/username" and copies it into "/Desktop" on your device. Example: beginlstlisting

```
$ scp -r username@hpc-logon.nyit.edu:/data/users/username/a_file.txt
    otherusername@hpc-logon.nyit.edu:/data/users/otherusername
```

copies a file named "a_file.txt" from the cluster account of "user" to that of "otheruser".The same logic applies to copying directories from the cluster to a device or from one cluster account to another.

2.1.4 Removing files, directories

The command "rm" is used to remove files within a directory whereas the command "rm -r" is used to remove whole directories. One has to be very cautious with these commands since deletion is final on the cluster. Example:

```
$ rm -r /data/users/username/a_folder
```

deletes the directory named "a_folder". **The -r option allows for a process to be performed in more than one file or directory.**

```
$ rm -r ADirectory BDirectory ##deletes both directories
```

2.1.5 Use of *

All the commands covered so far are able to perform their respective actions on more than one file or directory. Proper use of the * option can prove to be very effective. The asterisk is a chaining operator which performs a certain command on all the files that are similar to the argument provided, and if no argument is provided, the action is performed on all the files in a directory. Example:

```
$ rm -r a_dir* ##deletes all files and directories that start with a_dir
$ rm -r *.txt ## deletes all files that end with .txt
$ scp -r username@hpc-logon.nyit.edu:/data/users/username/*txt /Desktop ##copies
    all files that end in txt to the Desktop
```

2.2 Gcc/gfortran compilers, conda environment, jupyter notebook

Each cluster comes with several compilers installed, but to ensure that they are not missing, they could be installed with the following command:

```
$ sudo zypper in gnu-compilers-hpc
$ module load gnu
## Notice: administrator privileges are required to finalize the installing
```

By installing these compilers, the cluster is able to take high level coding languages (C++, fortran) and translate them to machine language. Python along with any other software, can be installed in the cluster in a similar way, by using the command $ sudo. However, access to the $ sudo command is given only to the cluster's administrative staff, as to not accidentally impede the cluster accounts of the other research staff. A user of the cluster without managerial priority can modify/update these software for personal use in almost any way desired using specific commands for each software installed, but not remove nor intall them.

2.2.1 Creating Conda environment

It is proper research discipline that whenever working on a project, one should create a specific environment for that. High scale (or even medium scale) programming projects require specific packages of a certain software to be installed and specific applications that interface only with those packages, hence it is a proper research discipline to create an environment for each project. Conda is an environment and package management system which can be used to create an environment using the following commands:

```
$ conda create --name my_env ## create environment named my_env
```

Using conda one can specify the packages to be downloaded on the environment. Example:

```
$ conda create -n my_env scipy ## download scipy package on this environment
```

Python is one of the main coding languages used in data science, and there many python distributions with a sufficient number of packages that could be downloaded on an environment, anaconda being one of the most useful of these distributions. To create an environment with anaconda:

```
$ conda create --name anaconda_env anaconda ## creates environment named
    anaconda_env with anaconda distribution in it
```

This environment is activated and deactivated with:

```
$ conda activate anaconda_env ## activate environment
$ conda deactivate ## deactivates environment
```

2.2.2 Jupyter notebook

Jupyter notebook is web-based platform that allows for computing in many languages, but it is mainly used for computing in Python. Jupyter notebook, same as the other packages/sofware, has to be downloaded with sudo access, and after the installation process is completed, a jupyter notebook can be created via the following:

```
$ jupyter notebook --no-browser --port=8888
```

On that command, "jupyter notebook" starts the notebook, "–no-browser" defines that the notebook can be accessed in any browser and "–port=8888" defines the port on the web where it can be found. Once this command has been run successfully, several lines of output will show up, and one of these lines will be a link in the format "http://localhost:4-digit-number/". The 4 digit number is the port number and that allows for access to the jupyter

notebook, and that number is taken as "8888" on the previous command so we will stick to that. In order to access the jupyter notebook from a local network device, a tunnel has to be created between the jupyter notebook and the device, this command has to be run from a command prompt terminal window:

```
$ ssh -t -t username@hpc-logon.nyit.edu -L 8888:localhost:8888 ssh node002 -L
    8888:localhost:8888
```

On the previous command, one could see that the node number is specified as "node002", and if that node is already in full use, SLURM will assign the notebook processing to another node. After that command is run and a tunnel is created with the device, the link provided as an oputput should be copied to any browser and accessed from there. If one wishes so, a password could be set to access the jupyter notebook for extra security. **Accessing jupyter notebook via the cluster allows for all the utility of this platform to be enjoyed at high computational speed.**