Accessing the NITD HPC

Using SSH in Windows

Windows systems do not have any built-in support for using SSH, so you will have to download a software package to use SSH. PuTTY is the most popular open source (ie free) SSH client for Windows. To install it, visit the download site, and download the Installer package. Once installed, find the PuTTy application shortcut in your Start Menu, desktop. On clicking the PuTTy icon The PuTTy Configuration dialog should appear: Locate the Host Name input box in the PuTTy Configuration screen. Enter the server name you wish to connect to (e.g. [username]@115.248.191.18), and click Open. Enter your password when prompted, and press Enter. You are now connected!

Using SSH in Mac or Linux

Both Mac and Linux systems provide a built-in SSH client, so there is no need to install an additional package. Simply locate and run the Terminal app. Once in the terminal, you can connect to an SSH server by typing the following command:

```
ssh [username]@[hostname]
```

For example, to connect to the HPC Login Node, with the username user1: ssh user1@ 115.248.191.18 You will be prompted for a password, and then will be connected to the server:

```
ssh user1@115.248.191.18
Last login: Wed Nov 6 08:41:31 2013 from 144.174.11.12
[user1@submit ~]$
```

Please generate an ssh key pair in your .ssh directory.

Copying Directory/File from local machine to HPC cluster:

To copy a local directory from your linux system (say InputFiles-2.0) to your home directory in your NITD cluster account, the procedure is:

1.from the terminal goto the parent directory using cd command.

```
user1@mylaptop3230:~$ cd ~/MyData/
```

2.under parent directory type ls <& press return key>, & notice InputFiles-2.0 is there.

```
user1@mylaptop3230:~$ls
MyFiles TempFiles-1.0 InputFiles-2.0
```

3.begin copy by typing:

```
user1@mylaptop3230:~$ scp -r InputFiles-2.0 (your username)@115.248.191.18:~
< you will be asked for password ; enter the same & press return key >
```

you should see :

```
scp -r InputFiles-2.0 user1@115.248.191.18:~
cover0.txt                100%   18KB  18.5KB/s   00:00
CPU.f                     100%   41    0.0KB/s   00:00
param0.txt                  100%   33KB  33.0KB/s   00:00
Final.exe                 100%  281KB 280.5KB/s   00:00
etc...
```

required files have been copied !

4.now login to your NITD hpc account as:

```
user1@mylaptop3230:~$ ssh (your username)@115.248.191.18
< you will be asked for password ; enter the same & press return key >
Last login: Thu Nov 26 16:13:31 2015 from 10.193.42.5
[user1@login03 ~]$
```

5.execute ls command, you should see InputFiles-2.0 directory.

6.execute

```
$ cd InputFiles-2.0
```

7.you can compile your fortran code using :

```
$ gfortran CPU.f
```

Follow those steps in reverse for transferring files from your NITD hpc account to your local directory. WinSCP (Windows Secure Copy) is a free and open-source SFTP, FTP, WebDAV and SCP GUI client for Microsoft Windows to transfer files between a local and a remote computer.

Enabling X11 Tunneling (graphical programs over SSH)

SSH, by default, allows you to run remote commands on a server in text mode. You can also use SSH to run graphical programs remotely, such as Matlab. On Linux and Mac clients, simply include the -X parameter when connecting via SSH:

```
ssh -X fsuid@spear-login.rcc.fsu.edu
```

This option will enable remote graphical programs, such as Scilab, Matlab or Paraview to run over SSH. For example, on the HPC system, after connecting with the -X parameter, you should be able to execute the xeyes command and see the xeyes graphical user interface. Windows does not include built-in support for SSH . Free software package called Xming can be used, also, a commercial alternative to Xming is MobaXTerm. Notice the < username >@login03 section of the last line. That indicates that you are now connected to the 'submit' server. To disconnect from the remote server, simply type exit or close the terminal window.

Submitting job to HPC

PBS(Portable Batch System):

Portable Batch System (or simply PBS) is the name of computer software that performs job scheduling. Its primary task is to allocate computational tasks, i.e., batch jobs, among the available computing resources.PBS is a distributed workload management system. PBS handles the management and monitoring of the computational workload on a set of one or more computers.

Please note that the login nodes are NOT to be used for long jobs. Any processes running on the login node would be killed by admins. To test, debug and to interactively run any programs, please request for a interactive job.

Basic PBS Commands:

JOB CONTROL

1.**qsub** is used to submit an executable script to a batch server. The script is a shell script which will be executed by a command shell such as sh or csh. e.g.

```
qsub -P cc myjobscript
```

It will return**job_ID.NITD1** i.e.**sequence-number.servername**. job_ID is the job identifer which can be further used for any actions involving the job, such as checking job status, modifying the job, tracking the job, or deleting the job. Here, cc is your department identifier, and myjobscript is a textfile conting instructions for the job.

some useful options

```
qsub      -q <queue> <jobscript>          # submit to specified queue
          -N <jobname>                    # set job name (default: name of jobscript)
          -l <resources>=<value>          # request for required resoures
                                  # (select,ncpus,ngpus,nmics,walltime,place etc.)

          -l select= <n>                   # request n number of nodes/slots
          -l ncpus= <n>                    # request n number of cpus on a node
          -l ngpus=  <n>                   # request n number of gpus per node
          -l nmics=  <n>                   # request n number of mics per node
          -l place=scatter                 # scatter the proceses across nodes
          -l place=pack                    # places the process across nodes in compact
                                             way
          -l host=<node name >             # run a job on a particular node

          -P <project Name>                # Specify the name of the project
          -o <Output>                      # for specifying the pbs output file name
          -e <Error>                       # for specifying the pbs error file name
          -M <email>                       # full email address for notification
```

```
 e.g. qsub -N test -lselect=20:ncpus=24 -P cc -o out -e err myjobscript
          #Will request for 20 node with 24 cpus each under project "cc" with
          output file name "out" and error file name as "err"


e.g. qsub -N test -lselect=2:ncpus=24:ngpus=2 -P cc -o out -e err myjobscript
          #Will request for 2 node with 24 cpus each with 2 mic cards on each node

e.g  qsub -N test -lselect=4:ncpus=24:nmics=2 -P cc -o out -e err myjobscript
          #Will request for 4 node with 24 cpus each with 2 mic cards on each node

e.g  qsub -N test -lselect=4:ncpus=12 -P cc -l place=scatter myjobscript
          #Will request for 4 node with 12 cpus each and will scatter the jobs
            across  4 nodes equally.

e.g  qsub -N test -lselect=1:ncpus=24:host=cn001 -P cc  myjobscript
          #Will request for 1 node with 24 cpus each but only run on node cn001
```

NOTE :

Interactive Jobs: If you want to run a job interactively use "**-I**".Will provide you the shell on the compute nodes to run the job interactively.

```
e.g. qsub -I -P cc -N test -lselect=2:ncpus=24
```

2.**qstat** is used to view job and job step information for jobs managed by PBS. The default view includes only minimal information.

```
qstat       -Q                      # show all queues
            -Q <queue>              # show status of specified queue
            -f -Q <queue>           # show full info for specified queue
            -q                      # show all queues (alternative format)
            -q <queue>              # show status of specified queue (alt.)
qstat       -a                      # show all jobs
            -a <queue>              # show all jobs in specified queue
            -f <job_ID>             # show full info for specified job
            -n                      # show all jobs and the nodes they occupy
            -T                      # show the estimated starting time of the Job
                                      (It is calculated in every two hours.)
            -x                      # show recently finished jobs
                                      (Job history is preserved for past 24 hours.)
e.g.    qstat -ans job_ID       # to check the details of the job
        qstat -u $USER          # to check the job submited by the user
        qstat -T job_ID         # shows the estimated starting time of the Job
```

NOTE:

-Presently there are two queues for users i.e. "standard" which is the default queue, and a low priority queue "low"

-Jobs status can be:

"Q": Queued. Job is queued and will start when all resource requirements are met.

"R" : Running. Job is currently running.

"F" : Finished (Accesible via qstat -x)

"H" : Held (Job is held and will NOT run)

Details of the state can be seen via

```
qstat -awns <jobid>
```

or

```
qstat -f <jobid>
```

3.**qdel** deletes jobs in the order in which their job identifiers are presented to the command. A job is deleted by sending a Delete Job batch request to the batch server that owns the job. A job that has been deleted is no longer subject to management by batch services.

A batch job may be deleted by its owner, the batch operator, or the batch administrator.

```
e.g. qdel job_ID             # to delete a job
```

4. **qalter** alters one or more PBS batch jobs. The attributes listed with the options to the qalter command can be modified. If any of the modifications to a job fails, none of the jobs attributes is modified.

```
e.g. qalter -l ncpus=4 job_ID   # change the no. of ncpus for the requsted job
     qalter -l select=1:ncpus=4:mem=512mb job_ID
```

NOTE :

-If a job is running, the only resources that can be modified are cput (CPU time) and walltime.

-If a job is queued, any resource mentioned in the options to the qalter command can be modified, but requested modifications must fit within the limits set at the server and queue for

the amount of each resource allocated for queued jobs.
-If a requested modification does not fit within these limits, the modification is rejected.
5. **qhold** requests that a server place one or more holds on a job. A job that has a hold is not eligible for execution. Supported holds: USER , OTHER (also known as operator), SYSTEM, and bad password.
A user may place a USER hold upon any job the user owns. An operator, who is a user with operator privilege, may place either a USER hold or an OTHER hold on any job. The batch administrator may place any hold on any job.
The p option can only be set by root or admin user via qhold -h p. The owning user can release with qrls -h p or query by qselect -h p.

e.g. qhold job_ID # place the job on hold

6.**qrls** removes or releases holds which exist on batch jobs.A user may always remove a USER hold on their own jobs, but only privileged users can remove OTHER or SYSTEM holds. An attempt to release a hold for which the user does not have the correct privilege is an error and no holds will be released for that job.

e.g. qrls job_ID # release user hold on the job

NOTE :

-The alloc manager hook sets a "user hold" on the jobs

e.g. qrls $(qselect -s H -u userid -q low) # users can release their job themselves

7.**qrerun** directs that the specified jobs are to be rerun if possible. To rerun a job is to terminate the session leader of the job and return the job to the queued state in the execution queue in which the job currently resides. If a job is marked as not rerunable then the rerun request will fail for that job.

e.g. qrerun job_ID # re-run the job

8.**qrun** is used to force a batch server to initiate the execution of a batch job. The job is run regardless of scheduling position or resource requirements.

e.g. qrun job_ID # run the job

NOTE :

-In order to execute qrun, the user must have PBS Operation or Manager privileges.

9. **tracejob** utility extracts job status and job events from accounting records, MOM log files, server log files, and scheduler log files. Using it can help identify where, how, a why a job failed. This tool takes a job id as a parameter as well as arguments to specify which logs to search, how far into the past to search, and other conditions.

e.g tracejob -n 10 job_ID #look for the job from last 10 days logs

10.**Dependancy Jobs** PBS allows you to specify dependencies between two or more jobs. Dependencies are useful for a variety of tasks, such as:

•Specifying the order in which jobs in a set should execute

•Requesting a job run only if an error occurs in another job

•Holding jobs until a particular job starts or completes execution

There is no limit on the number of dependencies per job.

Dependency	Definition
after	**Execute current job after listed jobs have begun**
afterok	**Execute current job after listed jobs have terminated without error**
afternotok	**Execute current job after listed jobs have terminated with an error**
afterany	**Execute current job after listed jobs have terminated for any reason**
before	**Listed jobs can be run after current job begins execution**
beforeok	**Listed jobs can be run after current job terminates without error**
beforenotok	**Listed jobs can be run after current job terminates with an error**
beforeany	**Listed jobs can be run after current job terminates for any reason**

The construct of a dependency i "**-**W depend=dependency expression".

e.g qsub -W depend=afterok:Job_ID jobscript
e.g qsub -W depend=after:Job_ID jobscript
e.g qsub -W depend=before:Job_ID jobscript

11.**qorder** changes the order of the jobs within a queue.The order two jobs is to exchange the job's position.

e.g qorder <Job_ID1> <Job_ID2> #Will switch the positon of the Jobs.

- For more info please visit: PBS Works Documentation
- You must submit yout job to a project that you are a part of. Everyone is part of a default project with the same name as their department dc name: the string that appears after /home/ in your home dierectory's path. For example a user (whose home path starts with "/home/cc/" will use qsub **-**P cc to specify her default project as cc.
- **Job limits:**

- Upper time limit: 168 hours (7 days)
- The entire cluster can be subscribed for a maximum of 24 hours
- A fraction of the cluster can be subscribed for a maximum of 24/fraction hours. e.g. half the cluster can be requested for 48 hours, quarter of the cluster can be requested for 96 hours etc.

Examples of resource selection:

1.24 CPU cores per node :

-l select=2:ncpus=24

2.2 GPUs per node :

-l select=2:ngpus=2

3.2 Xeon Phi Per node :

-l select=2:nmics=2

4.High memory (505GB):

-l select=1:highmem=1

5.Old K20 GPU:

-l select=1:ngpus=2:K20GPU=true

Typical submission script

In a blank text file (e.g. pbsbatch.sh) add the following: "#PBS" is NOT a comment! #PBS is an instruction to PBS.

```
#!/bin/sh
### Set the job name (for your reference)
#PBS -N Your_job_name
### Set the project name, your department code by default
#PBS -P cc
### Request email when job begins and ends
#PBS -m bea
### Specify email address to use for notification.
#PBS -M $USER@nitdelhi.ac.in
####
#PBS -l select=n:ncpus=m
### Specify "wallclock time" required for this job, hhh:mm:ss
#PBS -l walltime=01:00:00

#PBS -l software=replace_with_Your_software_name
# After job starts, must goto working directory.
# $PBS_O_WORKDIR is the directory from where the job is fired.
echo "==============================="
echo $PBS_JOBID
cat $PBS_NODEFILE
echo "==============================="
cd $PBS_O_WORKDIR
#job
time -p mpirun -n {n*m} executable
#NOTE
# The job line is an example : users need to change it to suit their
applications
# The PBS select statement picks n nodes each having m free processors
# OpenMPI needs more options such as $PBS_NODEFILE
```

---

submit using:

```
$ qsub pbsbatch.sh
```

Interactive job using department (project) cc

2 nodes, 24 procs each:

```
$ qsub -P cc  -I -l select=2:ncpus=24
```

2 nodes, 24 procs each, 2 mpitasks per node:

```
$ qsub -P cc  -I -l select=2:ncpus=24:mpiprocs=2
```

2 nodes, 2 gpus each, 2 cpus each (default) for 6 hours:

```
$ qsub -P cc  -I -l select=2:ngpus=2 -l walltime=6:00:00
```

2 nodes, 2 gpus each, 12 cpus each (explicit):

```
$ qsub -P cc -l select=2:ngpus=2:ncpus=12
```

Steps for preparing and submitting your job

How to load required software

MATLAB

You can load matlab via **module load** command

Load any one module from the below:

```
module load apps/Matlab/r2014b/gnu
module load apps/Matlab/r2015b/gnu
module load apps/Matlab/r2016b/precompiled
module load apps/Matlab/r2017a/precompiled
module load apps/Matlab/r2017b/precompiled
```

PYTHON

```
Python:

To load python 2.7.13 in your environment, please use -

compiler/python/2.7.13/ucs4/gnu/447

Please note that after executing previous command, the python packages
like numpy, scipy will not be available in your environment. After
loading python, you need to explicitly load an entire python package
suite as-

pythonpackages/2.7.13/ucs4/gnu/447/package_suite/1

Following command will help you with the list of available python
package within a "package_suite" -

module help pythonpackages/2.7.13/ucs4/gnu/447/package_suite/1

Or, you could selectively load only required modules from the output of
following command -

module avail pythonpackages/2.7.13
```