

Guide for running programs on HPC servers

Pradeep Kumar

12 Oct, 2011

High Performance Computing(HPC) can be used to run programs which are computationally expensive and take a lot of time to run. There are two ways to run programs on the HPC servers: interactive and non-interactive. Smaller programs are usually run in an interactive way and it is similar to the way we program on our desktops. Larger programs are usually run in a non-interactive way. By non-interactive I mean that you don't get to see the user interface of MATLAB or other softwares as you see on the desktop. You may use the following three steps to run programs on the HPC servers:

Step 1: Logging into the servers: You can use your university id and password to log in. All faculty and grad students have an account on the HPC servers as of August 2011. If there is a problem logging in, contact Brad Winters(baw128@psu.edu). There are two servers that will be of use:

lionxf.rcc.psu.edu - For running large programs non-interactively(in batch mode).

hammer.aset.psu.edu - For running small programs interactively.

To log into the lionxf server from your desktop you have to use the SSH secure shell client program(available on all department computers). Once you run the SSH client icon, click on add profiles and give it a name of your choice. It will then ask for host name(server), psu user id and password to log into the server.

There are two options for logging into the hammer server from your desktop. You can either use the above mentioned SSH secure shell client program or use the Windows Remote Desktop utility to log in. Windows Remote Desktop utility can be invoked from: Start Menu - All Programs - Accessories. Windows Remote Desktop provides a graphical interface to the linux server unlike the console interface provided by SSH. For more details see: http://rcc.its.psu.edu/user_guides/remote_connectivity/

After logging into the servers you need to enter linux commands to move through the file system, add directories, copy and save files, etc. Learning these commands is a small cost and it makes working more efficient in the long term. To learn the basic commands follow the links:

<http://freeengineer.org/learnUNIXin10minutes.html#Viewing>

<http://www.unixguide.net/linux/linuxshortcuts.shtml>

Once you log into the server you will land into the home directory: `/gpfs/home/<userid>`. There will be two subdirectories into this home directory: a work directory at `/gpfs/work/<userid>`, and a scratch directory at `/gpfs/scratch/<userid>`. Work directory is optimized for performance and has a limit of 64 GB. Data is backed up overnight into tapes in this directory. All the work should be done into the work directory. Home directory is used for keeping important document like draft of thesis or important programs. There are two independent backups for home directory and it has a limit of 8 GB. Scratch directory has over 130 TB of space available for anyone to use. Files here aren't backed up in this directory.

Step 2: Once you log into the server you will see a Linux screen. To run any software, you have to load it first. The general command to load a piece of software is: `module load <package name>`. For example,

For loading Matlab: `module load matlab`

For loading the compecon toolbox written for Matlab by Miranda and Fackler: `module load compecon`

For loading Mathematica: `module load mathematica`

For loading C++: `module load gcc`

These commands have to be executed to run on either machine(hammer or lionxf). To get a list of all available packages on the server use the command: `module avail`. Once you log out of the server, these modules will be automatically unloaded.

2 (a) Running a MATLAB job interactively on hammer: If you want to have an experience similar to the MATLAB on desktop, invoke the Windows Remote Desktop utility and follow the steps below:

1. Enter the computer name: hammer.aset.psu.edu. Login with your psu userid and password.
2. It will show the graphical interface of a linux desktop. Right click and select "open terminal".
3. In the terminal type: module load matlab
4. If you want to use any of the commands in the compecon toolkit then type: module load compecon
5. Next command should be: matlab. It will give you a graphical interface similar to what you see on your desktop.

Note: Graphical logging into the server is fast only from computers in the department or anywhere in the university with a wired connection. Running Windows Remote Desktop from computers at home on a wireless connection tends to be slow and unpleasant. Running SSH client is always fast but comes with the cost of working in a console mode.

2(b) Running a MATLAB job non-interactively: For running large programs you have to write certain commands in a file(a PBS script) and run this PBS script on the Linux server(lionxf server). This PBS script will work as a request for running a program on the HPC server. You can write this PBS script on the Linux server directly by using a VI/Nano/Emacs editor or you can write this file in notepad or other text editor on your desktop and FTP it to the lionxf server. For FTP, you can use the "SSH Secure File Transfer Client" program available on all department computers. Below is a sample PBS script to run a MATLAB program:

```
##### test.txt #####
# This is a sample PBS script. It will request 1 node for 200 hours, 10 gb of memory to run a matlab
# program named"code_5aug.m" which resides in the directory "work/banking"
# Request 1 processors on 1 node - Use more nodes if you are using parallel computing.
#PBS -l nodes=1
# Request 200 hours of walltime - Its the upper bound of time you think your program will take to finish.
#PBS -l walltime=200:00:00
# Request that regular output and terminal output go to the same file
#PBS -j oe
# Requests 10gb of memory -The default here is 1 gb which is sufficient for most of the jobs.
# Requesting higher memory can put you behind in the queue.
#PBS -l pmem=10gb
# Requests to run the program in the "lionxf-econ" queue which is allocated for our department.
#PBS -q lionxf-econ
# By default, PBS scripts execute in your home directory, not the directory
# from which they were submitted. The following line places you in the directory where
# you matlab program lies.
cd work/banking
echo " "
echo " "
echo "Job started on 'hostname' at 'date'"
module load matlab
unset DISPLAY
matlab -r code_5aug
echo " "
echo "Job Ended at 'date'"
echo " "
##### file ends #####
```

Some points to note about the above PBS script:

- Anything starting with # is a comment unless it is followed by the keyword PBS.
- Save the PBS script file (e.g. test.txt) in the home directory. But keep your MATLAB/C/C++ files under the work directory.

- *echo* command prints the starting and ending time of the program in the output file.

Step 3: Once you save the file on the Linux server, you can run it by using the following command on the Linux terminal:

```
qsub test.txt
```

Running this command will give you the following output:

```
12345.lionxf.rcc.psu.edu
```

This shows that your job has been submitted successfully and gives you a job id. In this case 12345 is the job id.

Some additional commands:

- If you want to know the status of your job, use the following command(Use your psu id e.g. puk125):

```
qstat -u psuid
```

The above command will tell you the time elapsed since your job started running. If it doesnt give any output, thats an indication that the program has finished running(with or without errors).

- You can delete a job by using the following command:

```
qdel 12345
```

The above command just needs the relevant job id after the keyword *qdel*.

When the job is finished all the output will be saved in a file named after the PBS script file. In the above example it will be saved in the file: test.txt.o12345 .

You can read more about running jobs on: http://rcc.its.psu.edu/user_guides/system_utilities/pbs/

Calling KNITRO from a C/C++ program

First, it is useful to know that you can find KNITRO at `/usr/global/knitro/`. Currently, there are three directories there: 6.0.1, 7.0.0, and current.

The last directory (current) is a symbolic link which references directory 6.0.1.

Directory 7.0.0 contains four important subdirectories:

- doc/ – here you can find KNITRO User Manual;
- examples/ – this directory contains examples of calling KNITRO from C, C++, Fortran, Java, and Matlab;
- include/ – this directory contains header file `knitro.h` which is needed to compile C/C++ programs with KNITRO;
- lib/ – this directory contains knitro libraries which are needed to link and run C/C++ programs with KNITRO.

A minimal working example of a C program with KNITRO in Eclipse IDE (on the hammer server):

1. In a terminal execute the command:

```
module load knitro
```

to load all the environment information needed for KNITRO (version 7.0.0).

2. From the same terminal launch Eclipse by executing:

```
module load eclipse
```

```
eclipse
```

During the start up, Eclipse will ask you to choose a working directory. This is a directory where you will store your Eclipse projects.

3. Create a new C project: File -> New -> Project -> C/C++ -> C Project. If Eclipse asks you switch to C/C++ Perspective, choose to do so.

4. Copy files `callbackExample1.c`, `knitro.opt`, `problemDef.h`, `problemHS15.c` from `/usr/global/knitro/7.0.0/examples/C` to the newly created project directory.

5. In order to build the project, you need to link two libraries with it: `knitro` and `dl`.

For that choose Project -> Properties -> C/C++ Build -> Settings -> Tool Settings -> GCC C Linker -> Libraries, and add `knitro` and `dl` to the list of libraries.

Note that you do not need to specify paths to these libraries. All the necessarily paths were added to corresponding environment variables as a result of loading of the `knitro` module.

6. Build the project by pressing `Ctrl + B`.

7. Run the project by pressing `Ctrl + F11`.

At this point you should be able to see the output of the project: results of solving the problem implemented in `problemHS15.c`.