A Guide on Hoffman2 Cluster & Linux Operating System

Jian Wang `wj@ucla.edu`
Physics and Astronomy, UCLA

August 29, 2018

hoffman2 is a Linux machine

$ `myquota` check the usage of your quota
$ `mygroup` see what type of resources you have available
$ `groupjobs` check group jobs
$ `qstat` check all the jobs in the supercomputer.

for more questions please check: https://www.hoffman2.idre.ucla.edu/faq/

compile and run c++

First of all, the c++ compile-link-run stereotype should work for any machine. Watch this video and my notes on wordpress to understand these procedures.

link 3rd party lib One of the difficulty is to link 3rd party c++ lib, for example, FFTW3 (fastest Fourier transform in the West :). There are two steps:

1. download and install the lib in your machine
2. link your code to the lib when compiling

Step 1, sometimes you don't know where the lib is installed,
in hoffman2, it says the location is `$FFTW3_HOME=/u/local/apps/fftw3/current`;
in my Yoga 260 laptop, I downloaded and the locations seem to be `/usr/local/bin` and `/usr/local/lib` , confused about the location, when I `$locate fftw` , I found several places contains fftw.

Step 2, add parameters to indicate the link
In my Yoga260 PC, (path already configured):

```
$ g++ -std=c++11 fftw_test.cpp -lfftw3 -o fftw_test.exe
```

The command line above will compile the code fftw_test.cpp into fftw_test.exe,
where the code `#include <fftw3.h>`
`-lfftw3.h` is the key word here
hoffman2, (path need to be included explicitly using key words `-I` and `-L`):

```
$ gcc -std=c99 pgm.c -I$FFTW3_HOME/include -L$FFTW3_HOME/lib -llib -lm [-static] -o pgm
```

depending upon the compiler you want to use.

- Replace pgm.c with the name of the file containing your source code
- Replace pgm with the name of the executable to be created
- Replace lib with one of the three fftw3 libraries: fftw3, fftw3f or fftw3l
- Set FFTW3_HOME /u/local/apps/fftw3/current

If you omit -static, you will have to set the LD_LIBRARY_PATH environment variable at run time to include `FFTW_HOME/lib`
You can either set an environment variable or replace `$FFTW3_HOME` in the command shown above (the above is copied from here, read for details)

Listing 1: compile.sh

```
#!/bin/bash
FFTW3_HOME="/u/local/apps/fftw3/current"
echo $FFTW_HOME
. /u/local/Modules/default/init/modules.sh
module load intel/13.cs
module load gcc/4.9.3
g++ -static -std=c++11 fftw_test.cpp -I$FFTW3_HOME/include ...
    -L$FFTW3_HOME/lib -lfftw3 -lm
```

without line 4, the computer doesn't know `module` in line 5 and 6. `-lm` is another math lib?

(to be confirmed) These keys words `-I` , `-L`, see my wordpress, are not necessary if `$PATH` already has been written.

run c++ program extensively in hoffman2

After compiling the code successfully, say, we have a program named `demo`, it takes parameters and can be launched like this in terminal:

```
$ demo para1 para2
```

What I wish to achieve is,

1. to submit this jobs in queue using UGE language
2. submit job arrays (using same program, but modify parameters)
3. organize the input, output informations in folders
4. other trick to overcome the limited running time

Read this guide by hoffman2 people and a better version by neuronscience people

demo codes

these demonstration codes in hoffman2 are store at location
`/u/home/w/wwwjjj/ABC`

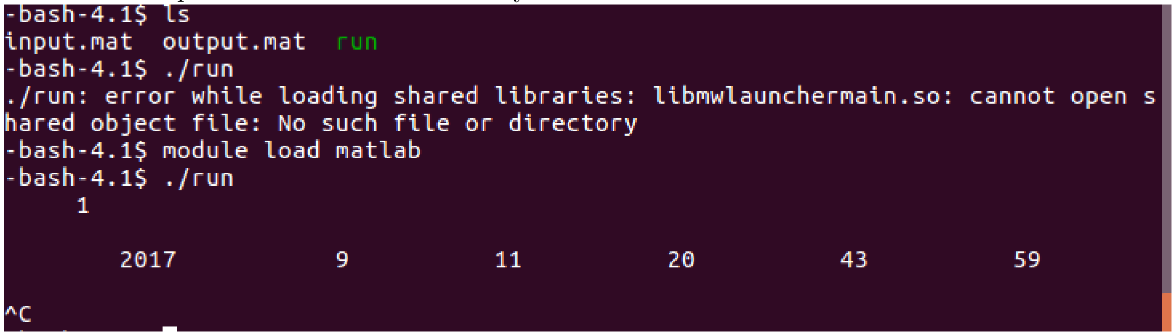
1. `std::getenv()` don't forget to `#include <cstdlib>`

compile and run Matlab

Matlab is a commercial software. The licenses are limited on Hoffman2, we can only run limited amount of programs using the software. The solution is to compile the Matlab codes to a standalone program first, and then run the program without licenses.

method 1 compile and run separately
`$ module load matlab`
`$ mcc -m file.m`
after this, there will be a file named `file`, run it.
`$ ./file`
or to use `$ matexe.q` to submit.

Before running the executable program, need to `$ module load matlab` , otherwise the computer couldn't find the library.



```
-bash-4.1$ ls
input.mat  output.mat  run
-bash-4.1$ ./run
./run: error while loading shared libraries: libmwlaunchermain.so: cannot open s
hared object file: No such file or directory
-bash-4.1$ module load matlab
-bash-4.1$ ./run
     1

        2017           9          11          20          43          59

^C
```

However, there are different versions of Matlab, if you `mcc` using one version, but run the program under other version, they conflict.

method 2 using the `application.queue` syntax
these are provided by the hoffman2 people
https://www.hoffman2.idre.ucla.edu/matlab_toolboxes/ :
`$ matlab.q` Runs single or multi-processor in two steps: compile and execute.
`$ mcc.q` Use the MATLAB compiler to create a stand-alone executable.
`$ matexe.q` Run a MATLAB stand-alone executable created with mcc

method 3 running matlab interactively
`$ matlab`

tips for Matlab in terminal

plot terminal don't have a window to display the graphs, the solution is to save the "magic.png" file, set visible off.

Listing 2: plot codes

```
h=figure('visible','off');
imagesc(magic(8));
print(h,'-dpng','magic');
close(h);
```

addpath addpath only works during compiling process, we don't need to run it during running. isdeployed returns true, if the program is standalone.e

Listing 3: addpath

```
if(isdeployed == false)
addpath('./B0_randfield/');
addpath('./B1_getH_XYE/');
addpath('./B2_two_wick/');
addpath('./B4_correlator');
addpath('./pfaffian/');
end
```

running Python

Git

`$ git init` enter a folder, then initialize the git.
`$ git clone git@github.com:<username>/<repository>.git` clone the codes

univa grid engine (UGE)

UGE is a system to schedule and control running jobs in supercomputer. It is like the Air traffic control in LAX, scheduling when and which airplane to take off, and to land. Knowing the grammar of UGE helps your codes to take off and run smoothly.

Example, submit a executable program using UGE:
Method 1, parameters in line:

```
$ qsub -cwd -o path -M login_id@mail -m bea -l h_data=1024M,h_rt=24:00:00 myjob.cmd
```

Listing 4: myjob.cmd

```
#!/bin/csh
/path/to/executable
```

Method 2: parameters in a file

```
$ qsub myjob.cmd
```

Listing 5: myjob.cmd

```
#!/bin/csh
/path/to/executable
#$ -cwd
#$ -o path
#$ -M login_id@mail
#$ -m bea
#$ -l h_data=1024M,h_rt=24:00:00
```

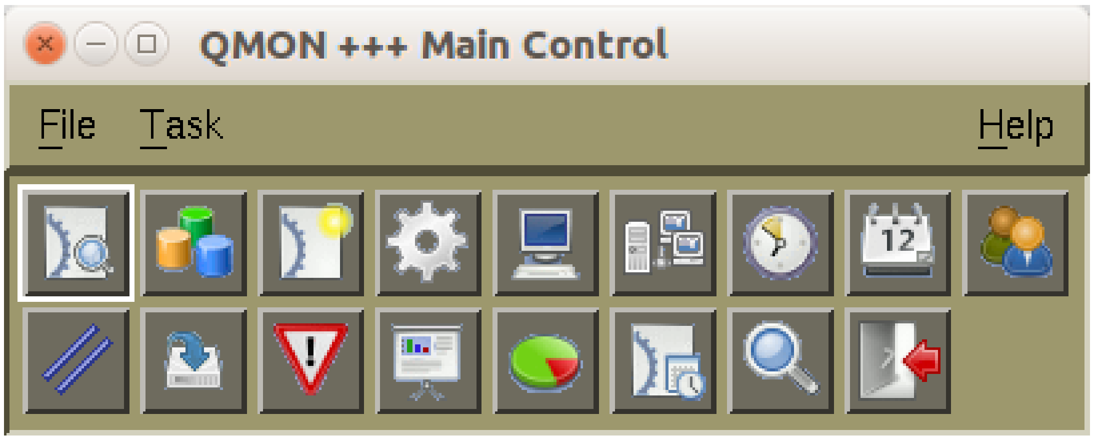
There are multiple ways to check who is competing the computer resource with you.
`$ qstat` , a command from UGE
`$ groupjobs` , a command made by the hoffman2 people

For details, please check: https://www.hoffman2.idre.ucla.edu/computing/sge/

For GUI view, ssh login with `-X` option, then try `$ qmon` command.

```
$ ssh -X username@hoffman2.idre.ucla.edu
$ qmon
```

There, you will see a window, these icons make command easier.



parallel & GPU

files

general guide on using hoffman2
https://www.hoffman2.idre.ucla.edu/computing/ hoffman2 people
https://idre.ucla.edu/sites/default/files/run-par-jobs.pdf?x83242 parallel guide
https://www.ccn.ucla.edu/wiki/index.php/Hoffman2 by some neuron scientists