AMS 200, Oct 20, 2014

Baskin Engineering UC SANTA CRUZ

AMS 200: Working on Linux/Unix Machines

Prof. Dongwook Lee (dlee79@ucsc.edu)

Department of Applied Mathematics and Statistics
University of California, Santa Cruz

1. Remote login via SSH

First of all, you will need to install an SSH (secure shell) client in order to access one of cluster machines (i.e., computing resources such as grape) remotely.

* If you're a PC user, you can download PuTTY from (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html).

* If you're a Mac or Linux user, you can simply use a terminal that is already available for you. For example, in Mac, go to **Applications** → **Utilities** and open **Terminal** application. In Linux, **Terminal** can be found under **System** in general.

Next step is to log in yourself to one of the AMS machines. For today, you are logging in to a cluster named "grape". To log in to the cluster, you need to use a command `ssh` using the terminal we just mentioned. In the command line, you type in

```
ssh -X your_name@grape.soe.ucsc.edu
```

with your SOE login password. At the step, you're logging into a master node or login node. As you login for the first time to the master node, you are asked to enter a passphrase. You can enter a very secured if you wish, or you can simply press enter. This process is to generate, so-called, "SSH keys", which is a way to identify trusted computers, i.e., the rest of compute nodes, without having to enter password every time you run parallel jobs on these compute nodes. Note that running a parallel job means that you run multiple jobs on multiple processors, which actually require you to login to the requested compute nodes with password. This SSH key generation save you from doing this process.

Your login is successful if you see something like the following on your terminal:

```
dongwook@grape.soe.ucsc.edu's password:
Last login: Sun Oct 19 21:06:33 2014 from c-50-131-18-210.hsd1.ca.comcast.net


  Computer technical support requests can be submitted via the web at
    https://itrequest.ucsc.edu

  or by e-mailing
    help@ucsc.edu

************************************************************

Online documentation for the grape cluster can be found at
http://grape.soe.ucsc.edu

Grape cluster is using the infiniband fabric.

************************************************************

Some Torque commands

qsub      --> Submits a job (create a shell script, then run qsub shellscript)
              use the -q option to specify which queue to use
qdel      --> Delete a job
qstat     --> see the status of jobs in the queue
qstat -Q  --> List of usable queues
pbsnodes  --> List status of all compute nodes


************************************************************

There are currently 4 queues on Grape

orig   -  compute-0-0-compute-0-4  PowerEdge 1950
          Intel(R) Xeon(R) CPU 2.33GHz 15G MEM

new    -  compute-0-5-compute-11  Dell PowerEdge R610
          Intel(R) Xeon(R) CPU 2.40GHz 15G MEM

newest  - compute-0-12-compute-0-19  PowerEdge R420
          Intel(R) Xeon(R) CPU 2.30GHz 32G MEM

default - includes all of the nodes.  This is the
          default queue.

************************************************************
```

2. Basic Linux Commands

There are few rules in using command lines in Linux. Several important rules are

* Commands are case-sensitive.

* Make sure you always logout yourself by typing `exit` when you're done.

* The Linux command lines enables you to create complex functions by combining built in command lines together. This capability gives you countless ways to make your commands work in various different ways.

Exercise 1: Please run matlab by typing in a command `matlab` on a command prompt.

Exercise 2: Please exit your current session and try to login again without having `-X` option. Please run matlab again. Is there any difference from the case with `-X`? You can use `-Y` instead of `-X`.

Answer to 1 & 2: `-X` or `-Y` option enables X11 forwarding in SSH, which provides you not only a command line interface from the console, but also a variety of graphical-user-interfaces (GUIs). With the X forwarding option in login, you can enjoy a full set of interface display functions remotely.

Here you're introduced to learn very basic Linux command lines. For more comprehensive studies, you can use to display a manual page using the `man` command, or you can come to ask the instructors for more help and resources.

2.1. Managing Files

> `ls` — lists your files
> `ls -l` — ls in long format
> `ls -a` — ls all files
> `mv filename1 filename2` — remane filename1 to filemane2
> `mv filename1 dirname` — move filename1 to a directory called dirname
> `cp filename1 filename2` — copy filename1 to filename2
> `rm filename` — remove a file
> `more filename` — display the contents of a file as much as will fit on your screen
> `less filename` — similar to `more` with the extended navigation capability allowing both forward and backward navigations
> `wc filename` — tells you number of lines, words and characters in a file
> `touch filename` — creating an empty file (multiple filenames after `touch` command will create multiple empty files)

Exercise 3: See if you can find `ls -l` and `ls -a` when you execute `man ls`.

2.2. Managing Directories

> `mkdir dirname` — create a new directory called dirname
> `cd dirname` — change directory, meaning you go to a directory called dirname
> `pwd` — tells you where you currently are in the directory tree
> `rmdir` — an empty directory deletion

Exercise 4: Create a directory called `dirA`, then under `dirA`, create an empty file named `filenameNull`.

Exercise 5: Delete the file `filenameNull`. Also delete the directory `dirA` using `rm` command. *Hint: Please look up man page of* `rm` *and find a useful*

option for directory deletion.

3. Editors

The following text editors are available on grape. You can choose whichever you want to use.

(1) vim (or vi):
To start — `> vi filename1`
To edit — enter "i" and start inserting text until <Esc> hit
To delete single character — `x`
To delete entire current line — `dd`
To exit — pressing <Esc> key followed by `:x <Return>` or `:wq <Return>` will quit `vi` saving the content to filename1, whereas pressing <Esc> key followed by `:q! <Return>` will quit `vi` without saving the latest change to filename1

See more basic commands for `vi` in http://www.cs.colostate.edu/helpdocs/vi.html

(2) emacs:
To start — `> emacs -nw filename1`
To edit — unlike `vi`, you can type in any characters in the editor
To save the current buffer– press hold down Control key and type in "x" and "s"
To save the current buffer with different file name — press hold down Control key and type in "x" and "w" and then enter new file name To exit the buffer — press hold down Control key and type in "x" and "c"

See more basic commands for `emacs` in http://www.cs.colostate.edu/helpdocs/emacs.html

4. Parallel Computing

Please read the pdf file by Prof. Nic Brummell:

https://ams.soe.ucsc.edu/sites/default/files/AMS_cluster_grape_evennewer_0.pdf

5. Two Examples

In both tests, you demonstrate one of the most basic tasks such as printing "hello world".

5.1. Running a short program in R – Statistic Computing

Use your preferred editor to implement the following lines and save it to a file named "test.R":

```
print(date())
set.seed(43678)
print("Hello World")
print("Random number from standard normal")
print(rnorm(1, 0,1))
```

There are two ways to run your R script. If you want to run the program using an R batch command, type from the Linux shell:

```
> R CMD BATCH test.R
```

Or, you can also run it by first start R (by typing "R" at the Linux command prompt) and once inside the R program, you can execute your source file with the command:

```
> source('test.R')
```

In the first approach, you can see your results written out to a default output file `test.Rout`. In the second example, you should be able to see the output on your screen such as

```
[1] "Mon Oct 20 00:43:37 2014"
[1] "Hello World"
[1] "Random number from standard normal"
[1] 0.7932329
```

5.2. Running a short program in Fortran – Scientific Programming Language

Again, please use your preferred text editor and implement the following short fortran program:

```
program hello
real ::  n,m
integer ::  i,j
i = 10
j = 2014
n=real(i)
m=159.e0
print *,"i+j=",i+j
print *,"n-m=",n-m
print *, "Hello World"
end program hello
```

You save it to hello.f90. Now you are going to compile it in order to generate an executable binary. On grape, you can do this using gfortran compiler:

```
> gfortran hellow.f90
```

After compiling your program, you should be able to see an executable binary file with a default name, `a.out`. Run it by entering a command line:

```
> ./a.out
```

Exercise 6: What does your result look like from running hello.f90?

Exercise 7: Can you give a different name for the executable instead of the default `a.out`?

Exercise 8: Can you extend the previous serial run of test.R (and/or hello.f90) to a parallel job, say 2 nodes with 8 processors on each node (i.e., a total of 16 parallel tasks)?