Tutorial: Linux commands and MORU Rocky Cluster

sompob@tropmedres.ac
MORU Mathematical Modelling Team
Mahidol Oxford Tropical Medicine Research Unit

7 June 2016, Version 0.6

1 Connect to Rocky Cluster

1.1 Secure Shell(SSH)

Secure Shell (ssh) is a network protocol similar to telnet that allows data to exchanged using a secure channel between your computer and the server. To use ssh, you need a ssh client program. PuTTY is recommended for Windows users. You can download PuTTY from `http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe`.

To connect to Rocky cluster,

- Run PuTTY.exe (just double click its icon.)
- Choose SSH protocol
- Enter "rocky" or "10.0.1.10" for the host name.
- Enter username and password.
- You will be asked some questions related to secure shell at the first time of your login. Just hit Enter key to go through them.

If you cannot connect to the cluster, please contact `helpdesk@tropmedres.ac`.

1.2 VNC

VNC (Virtual Network Computing) is a graphical desktop sharing system that uses the RFB protocol to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical screen updates back in the other direction, over a network. Therefore, you can run programs on a cluster in graphics mode from your computer. VNC consists of "server" and "viewer" programs. The "server" runs on the cluster and the "viewer" runs on your computer. You can download VNC viewer free version from `http://www.realvnc.com`.

To run VNC,

- Connect to the cluster in text mode using ssh.
- Start a vncserver process on the cluster by typing `vncserver` and then press Enter.
- The first time you run vncserver, it will ask you to set up a password. This is the VNC password that can be different to your login password. Once you have set your password, run `vncserver` again and you should see the output similar to this:

```
$ vncserver
You will require a password to access your desktops.

Password:
Verify:
xauth:  creating new authority file /home/Sompob/.Xauthority
```

```
New 'rocky.wellcome.or.th:1 (Sompob)' desktop is rocky.wellcome.or.th:1

Creating default startup script /home/Sompob/.vnc/xstartup
Starting applications specified in /home/Sompob/.vnc/xstartup
Log file is /home/Sompob/.vnc/rocky.wellcome.or.th:1.log
```

- vncserver gives you the display name, such as "rocky.wellcome.or.th:1".
- You can change your VNC password by running `vncpasswd`.
- Run the vncviewer program on your computer. Enter the display name, such as "rocky.wellcome.or.th:1",and then enter the VNC password. VNC will bring the desktop environment of the cluster to your computer screen.
- When you finish,
  - close the VNC viewer window on your PC.
  - stop vncserver on the cluster by running

    ```
    vncserver -kill :<displaynumber>
    ```

    For example,

    ```
    $ vncserver -kill :1
    Killing Xvnc process ID 30251
    ```

By default, VNC creates a startup file `~/.vnc/xstartup` that launches a basic window manager program called `twm`. If you would like to use another window manager such as GNOME (recommended), you have to change last line of `~/.vnc/xstartup` from `twm &` to `exec gnome-session &`. After changing, it looks like this:

```
#!/bin/sh

# Uncomment the following two lines for normal desktop:
# unset SESSION_MANAGER
# exec /etc/X11/xinit/xinitrc

[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig -iconic &
xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
exec gnome-session &
```

1.3 File Transfer

To transfer data between your computer and Rocky cluster, you need a "secure FTP" (SFTP) client program. If you use Windows, WinSCP is an open source freeware SFTP client that you can download from `http://winscp.net`. Cyberduck (`http://cyberduck.ch/`) is recommended for Mac user.

2 Basic Linux Commands

2.1 Getting Started: try a few simply commands

- `whoami` Print the current user id(user name)
- `date` Display the date and time
- `pwd` Print current working directory
- `finger` Print the online users
- `man` Show the manual of a linux command.

2.2 Password Changing: passwd

If you would like to change your password, use `passwd` command. As per the MORU IT policy, your password should be at least 9 characters minimum, mixed case, at least one special characters and one number, and ensure the password does not include any part of the username or is made using simple dictionary words.

```
[Sompob@rocky ~]$ passwd
Changing password for user Sompob.
Changing password for Sompob
(current) UNIX password:
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

2.3 List contents of the working directory: `ls`

`ls` displays the names of files and subdirectories in your working(current) directory.

```
[Sompob@rocky ~]$ ls
Acpypi  bin  namd-TK  PsN-3.1.0.tar.gz  SCmath  TEST  test-namd
```

If you add the option `-l` or just type `ll`, it will show the list of files and directories in more details.

```
[Sompob@rocky ~]$ ls -l
total 3852
drwxr-xr-x  4 Sompob users    4096 Dec 17 02:47 Acpypi
drwxr-xr-x  2 Sompob users    4096 Dec 18 00:23 bin
drwxr-xr-x  3 Sompob users    4096 Jan 18 17:37 namd-TK
-rw-r--r--  1 Sompob users 3913744 Feb  8 16:59 PsN-3.1.0.tar.gz
drwxr-xr-x 11 Sompob users    4096 Dec 24 03:10 SCmath
drwxr-xr-x 10 Sompob users    4096 Feb  8 16:32 TEST
drwxr-xr-x  2 Sompob users    4096 Nov 12 03:47 test-namd
```

The meaning of the output is that the first column is empty for data and programs or it is written character `d` for directory. The next three columns are permissions for the user(owner), the column 5,6,7 for the users in the user's group (here the group name is users.), and the last three for the others (not you and the members of your group). r = read (that file or directory can be read), w = write (It can be written or edited), x = execute (if it is a program, it can be executed or run).

2.4 Change the access permissions, owner and group

If you would like to change the permissions of a file or a directory, use `chmod` command. The permissions (r,w,x) are divided into three parts: owner(user, u), group(g), and other(o). The format of the `chmod` command is

```
chmod u(+/-)rwx,g(+/-)rwx,o(+/-)rwx target(file,directory)
```

For example, if you want the file `PsN-3.1.0.tar.gz` to be able to be read/written exclusively by you only, you can do that by typing

```
[Sompob@rocky ~]$ chmod u+rw,og-rwx PsN-3.1.0.tar.gz
[Sompob@rocky ~]$ ll
total 3852
drwxr-xr-x  4 Sompob users    4096 Dec 17 02:47 Acpypi
drwxr-xr-x  2 Sompob users    4096 Dec 18 00:23 bin
drwxr-xr-x  3 Sompob users    4096 Jan 18 17:37 namd-TK
-rw-------  1 Sompob users 3913744 Feb  8 16:59 PsN-3.1.0.tar.gz
drwxr-xr-x 11 Sompob users    4096 Dec 24 03:10 SCmath
drwxr-xr-x 10 Sompob users    4096 Feb  8 16:32 TEST
drwxr-xr-x  2 Sompob users    4096 Nov 12 03:47 test-namd
```

or if you would like only you and the members in your group to be able to read and write PsN-3.1.0.tar.gz, just type

```
[Sompob@rocky ~]$ chmod u+rw,g+rw,o-rwx PsN-3.1.0.tar.gz
[Sompob@rocky ~]$ ll
total 3852
drwxr-xr-x  4 Sompob users    4096 Dec 17 02:47 Acpypi
drwxr-xr-x  2 Sompob users    4096 Dec 18 00:23 bin
drwxr-xr-x  3 Sompob users    4096 Jan 18 17:37 namd-TK
-rw-rw----  1 Sompob users 3913744 Feb  8 16:59 PsN-3.1.0.tar.gz
drwxr-xr-x 11 Sompob users    4096 Dec 24 03:10 SCmath
drwxr-xr-x 10 Sompob users    4096 Feb  8 16:32 TEST
drwxr-xr-x  2 Sompob users    4096 Nov 12 03:47 test-namd
```

2.5 Make, Remove and Change Directory

- `mkdir` *dir* makes or creates a new directory *dir*.
- `rmdir` *dir* removes the directory *dir*. It will only remove it if it is empty.
- `cd` *dir* changes you working directory to the directory *dir*.

2.6 Copy file: `cp`

- `cp` *file1 file2* copies *file1* to a new file *file2*. If *file2* already exists, it is replaced.
- `cp -i` *file1 file2* copies *file1* to a new file *file2*. If *file2* already exists, it will ask if you want to overwrite it.(The `-i` stands for interactive.)
- `cp` *file1 dir* copies *file1* to a file with the same name inside directory *dir*.
- `cp` *file1 file2 dir* copies *file1* and *file2* to files with the same names inside directory *dir*. (Any number of files can be copied at the same time by extending the list file1 file2 file3...)

2.7 Rename and/or Move file: `mv`

- `mv` *file1 file2* renames *file1* as *file2*. If *file2* already exists, it is replaced.
- `mv -i` *file1 file2* renames *file1* as *file2*. If *file2* already exists, it will ask if you want to overwite it.
- `mv` *file1 dir* moves *file1* to a file with the same name inside directory *dir*.
- `mv` *dir1 dir2* has two possible results. If *dir2* does not already exist, it will change the name of the directory *dir1* to *dir2*. If *dir2* already exists it will move *dir1* (and its contents) to inside directory *dir2*.

2.8 Remove file: `rm`

- `rm` *file* will remove *file*.
- `rm -i` *file* will first ask whether you really want to remove *file*. It will only remove it if you type `y`.

2.9 Compressing and archiving files: `gzip, gunzip, tar`

To compress a file do

`gzip` *file*

This will replace *file* by a compressed version called *file*`.gz`. To recover a `.gz` file so that you can use it again, do

`gunzip` *file*.gz

Figure 1: Our MORU Rocky cluster.

When transferring a number of files by using `ftp, sftp` or email, it is easier to create a single file called a tar file which contain all the files. The files can then be extracted when the tar file reach its destination. To save space and make the transfer quicker, tar file are normally compressed using `gzip`. To create a gzipped tar file called *file*`.tar.gz`, type

`tar cvfz` *file*`.tar.gz` *list of files directories*

For example,

```
tar cvfz test.tar.gz TEST/ abcd.txt
```

would create a gzipped tar file which contains the file `abcd.txt` and the directory `TEST` and its entire contents (including further directories inside it). When the tar file is unpacked, it will automatically create the directory `TEST` if it does not already exist. To only see the contents of a tar file without extracting any file, type

`tar tfz` *file*`.tar.gz`

To extract the contents of a tar file, type

`tar xvfz` *file*`.tar.gz`

This command will overwrite old files with the same name as files in the tar file. Remove the `z` from the options and `.gz` in the name in the above when using tar files that have not been gzipped.

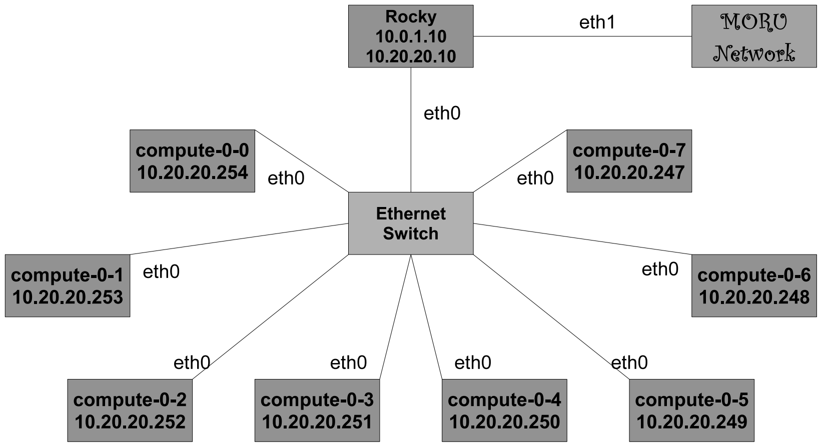

Figure 2: Network diagram of MORU Rocky cluster.

3 Rocky Cluster

3.1 Writing programs

3.1.1 Text editors

There are three text editors installed on the cluster.

- `nano` - Nano's ANOther editor, an enhanced free Pico clone
- `vi` - Vi IMproved, a programmers text editor
- `emacs` - GNU Emacs

3.1.2 Computer Languages

The following are the compilers installed on our cluster.

- `gfortran` - GNU Fortran 95 compiler
- `gcc` - GNU C and C++ compiler
- `bcc` Bruce's C compiler
- `g++` - GNU C++ compiler
- `perl` - Practical Extraction and Report Language
- `python` - an interpreted, interactive, object-oriented programming language

3.1.3 Compiling and running programs

To compile a C program called `xxx.c` and call the executable `xxx` type in

```
gcc -o xxx xxx.c
```

Assume you have a program written in C/C++ for solving the equations that govern the Lorenz oscillator in a file called `lorenz.c`.

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y-x)\\ \frac{dy}{dt} &= x(\rho-z)-y\\ \frac{dz}{dt} &= xy-\beta z\end{aligned}$$

```
//--------------lorenz.c-------------------------------
#include <stdlib.h>
#include <stdio.h>
#define  SIZE   10      //maximum number of DEs

double X,Y[SIZE],Y1[SIZE];
double xl,x0,h;
int    finesse,i,k,kl,l,n;

void F(double x, double *y, double *yp) {
     double sigma=10.,beta=2.66666,rho=28;
//y[0]=x, y[1]=y, y[2]=z
        yp[0] = sigma*(y[1]-y[0]);
        yp[1] = y[0]*(rho-y[2])- y[1];
        yp[2] = y[0]*y[1] - beta*y[2];
}

void RK4n(int n, double x, double *y, double h, double *y1) {
    double c1[SIZE],c2[SIZE],c3[SIZE],c4[SIZE],yy[SIZE],h2;
    int i;
    F(x,y,c1);
    h2=h/2.0;

    for (i=0; i<n; i++) yy[i]=y[i]+h2*c1[i];
    F(x+h2,yy,c2);

    for (i=0; i<n; i++) yy[i]=y[i]+h2*c2[i];
    F(x+h2,yy,c3);

    for (i=0; i<n; i++) yy[i]=y[i]+h*c3[i];
    F(x+h,yy,c4);

    for (i=0; i<n; i++)
        y1[i]=y[i]+h*(c1[i]+2.0*c2[i]+2.0*c3[i]+c4[i])/6.0;
}

int main(int argc, char *argv[])
{
  n=3;           // size of DE system
  x0=0.0;        // starting x
  xl=10.0;        // ending x
  kl=50000;         // number of calculated points
  finesse=100;
  h=(xl-x0)/kl/finesse;  // elementary integration step
  X=x0;
//  for (i=0; i<n; i++) Y[i]=1.0;  // starting Y values
    Y[0] = -10.;
    Y[1] = -12.0;
    Y[2] = 30.05;
  // write header
  printf("\n");
  printf(" ----------------------------------------------------------------------\n");
  printf("     X         Y1 <------------------------> YN estimated\n");
  printf(" ----------------------------------------------------------------------\n");
  // integration loop
  RK4n(n,X,Y,h,Y1);
  printf("%lf\t", X);
  for (i=0; i<n; i++) printf("%lf\t", Y[i]);
  printf("\n");
  for (k=1; k<=kl*finesse; k++) {
    X += h;
    for (i=0; i<n; i++) Y[i]=Y1[i];
    RK4n(n,X,Y,h,Y1);
    if (k%finesse==0) {     //Tabulate point
      printf("%lf\t", X);
      for (i=0; i<n; i++) printf("%lf\t", Y[i]);
      printf("\n");
    }
  }
  printf(" ---------------------------------------------------------------------\n");
  printf("\n");
  return 0;
}
```

You can compile this program by typing

```
gcc -o lorenz.exe lorenz.c
```

This will create an executable file called `lorenz.exe` from the input file `lorenz.c`. If your program uses mathematical functions from `math.h` you will need to add the `-lm` option into the above command. To run the program you simply type the name of the executable. For example, after compiling `lorenz.c` as above you would then type

```
lorenz.exe
```

or if "." is not in your path (you can see your path by typing `echo $PATH`)

```
./lorenz.exe
```

to run it. If you want to stop a program run in this way before it completes its task, press `<Ctrl>-C`. In this example, `lorenz.exe` will generate the list of numbers and print it on screen. If you want to save this list for plotting in the program such as gnuplot (`http://www.gnuplot.info/`) or Mathematica (`http://www.mathematica.com/`) type

```
./lorenz.exe > data.dat
```

3.1.4 MPI programs

You can compile your C/C++ with MPI library using `mpicc, mpic++` by typing

```
mpicc -o outfile.exe sourcefile.c
```

or

```
mpic++ -o outfile.exe sourcefile.cpp
```

```
//-----------greeting.cpp---------------------------------
#include <iostream.h>
#include <math.h>
#include <mpi.h>

int main(int argc, char ** argv){
  int mynode, totalnodes;

  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD, &totalnodes);
  MPI_Comm_rank(MPI_COMM_WORLD, &mynode);

  cout << "Hello world from processor " << mynode << " of " <<
```

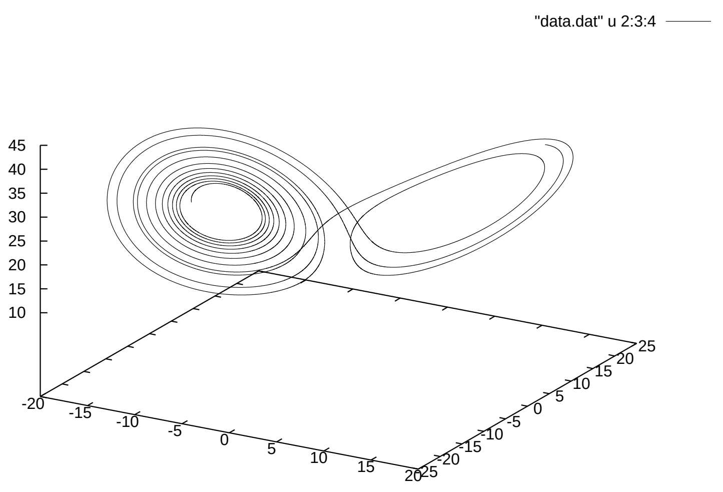

Figure 3: The graph of the results from lorenz.exe. It's plotted by gnuplot with the command "`splot ''data.dat'' u 2:3:4 with lines`".

```
totalnodes << endl;

  MPI_Finalize();
}
```

You can compile above C++ codes with mpi (greeting.cpp) by typing

```
mpic++ greeting.cpp -o greeting.exe
```

To run the mpi program `prog`, type

```
mpirun -np X prog
```

, where `X` is the number of CPUs and `prog` is the name of your program.

```
[Sompob@rocky greeting]$ mpirun -np 5 greeting.exe
Hello world from processor 1 of 5
Hello world from processor 4 of 5
Hello world from processor 0 of 5
Hello world from processor 2 of 5
Hello world from processor 3 of 5
```

3.2 How to submit batch jobs

3.2.1 Creating a job file

Here is an example of a job file called `greeting.sh`:

```
#!/bin/sh
#
# EXAMPLE MPICH SCRIPT FOR SGE
#
# Name of the job
#$ -N mpigreeting
#
# pe request for MPICH. Set your number of processors here.
# Make sure you use the "mpich" parallel environment.
#$ -pe mpich 4
#
# Set to working directory to current directory
#$ -cwd
#
# Combine stderr and stdout to one file
#$ -j y
#
# Run job through bash shell
#$ -S /bin/bash
#
# Adjust MPICH procgroup to ensure smooth shutdown
export MPICH_PROCESS_GROUP=no
#
# Print useful information
echo "Program start at: `/bin/date`"
echo "Got $NSLOTS slots."
echo "Machines:"
cat $TMPDIR/machines

# Run program. Should use full path instead of relative path
/opt/openmpi/bin/mpirun -np $NSLOTS /home/Sompob/example/greeting/greeting.exe

echo "Program finish at: `/bin/date`"
```

3.2.2 Submit the job file

To submit your job file to the cluster type

`qsub` *jobfile*

For example,

```
[Sompob@rocky greeting]$ qsub greeting.sh
Your job 2196 ("mpigreeting") has been submitted
```

In this exaple, your job ID is 2196. You will see the output files *mpigreeting.o2196*, *mpigreeting.po2196* created in your current directory. The output file .po???? (???? is the job ID.) will give you the list of the nodes which were used in the calculation and .o???? file is the output that your program printed on screen.

```
[Sompob@rocky greeting]$ more mpigreeting.po2196
-catch_rsh /opt/gridengine/default/spool/compute-0-1/active_jobs/2196.1/pe_hostf
ile
compute-0-1
compute-0-1
compute-0-2
compute-0-2
[Sompob@rocky greeting]$ more mpigreeting.o2196
Program start at: Mon Feb 15 12:00:45 ICT 2010
Got 4 slots.
Machines:
compute-0-1
compute-0-1
compute-0-2
compute-0-2
Hello world from processor 0 of 4
Hello world from processor 1 of 4
Hello world from processor 2 of 4
Hello world from processor 3 of 4
Program finish at: Mon Feb 15 12:00:45 ICT 2010
```

If you would like to submit your job to specific compute node, use `-l hostname` option. For example,

```
 qsub -l hostname=compute-0-1 xxx.sh
```

this would submit your xxx.sh job to the compute node, compute-0-1.

3.2.3 Check the status of the job queue

To check the status of your jobs on the cluster, type

- `qstat`

  ```
  [Sompob@rocky greeting]$ qstat
  job-ID  prior   name       user    state submit/start at     queue   slots ja-task-ID
  ---------------------------------------------------------------------------------------
     2134 0.00000 mpigreetin Sompob    qw    01/18/2011 15:57:01           4
  ```

- `qstat -f`

  ```
  [Sompob@rocky greeting]$ qstat -f
  queuename                      qtype resv/used/tot. load_avg arch          states
  ---------------------------------------------------------------------------------
  all.q@compute-0-0.local        BIP   0/0/32          0.00     lx26-amd64
  ---------------------------------------------------------------------------------
  all.q@compute-0-1.local        BIP   0/0/32          0.00     lx26-amd64
  ---------------------------------------------------------------------------------
  all.q@compute-0-2.local        BIP   0/2/32          0.03     lx26-amd64
  ```

- `qstat -f -u \*`

  ```
  queuename                      qtype resv/used/tot. load_avg arch          states
  ---------------------------------------------------------------------------------
  all.q@compute-0-0.local        BIP   0/1/32          1.02     lx26-amd64
     4435 0.55500 execute_ru Palang       r     04/28/2010 12:49:46     1
  ---------------------------------------------------------------------------------
  all.q@compute-0-1.local        BIP   0/2/32          2.01     lx26-amd64
     4440 0.55500 execute_ru Praiya       r     04/28/2010 13:56:31     1
     4457 0.55500 execute-ru Joel         r     04/30/2010 10:30:01     1
  ---------------------------------------------------------------------------------
  all.q@compute-0-2.local        BIP   0/1/32          1.00     lx26-amd64
     4438 0.55500 execute_ru Palang       r     04/28/2010 13:19:16     1
  ```

3.2.4 Delete your running job

To delete your running job, type

```
qdel  JobID
```

JobID is the numbers you get after submit your job with the command `qsub`.

3.3 NONMEM and PsN

We have NONMEM 7.2 (`http://www.icondevsolutions.com/nonmem.htm`) and PsN 3.6.2 (`http://psn.sourceforge.net/`) installed on the cluster (at `/share/apps/nm72` and `/share/apps/PsN-3.6.2` respectively). The followings are the example of the job file and the commands for submitting your jobs to the cluster.

3.3.1 `execute`

If you would like to run "execute" command with only one model file (.mod), the following is the example of the job file for submitting to the cluster.

```
#!/bin/sh
#
echo "Program start at: '/bin/date'"
#
# Name of the job
#$ -N execute-test
#
# Run program. Should use full path instead of relative path
/share/apps/PsN3.6.2/execute /home/Sompob/POPPK/test.mod -dir=test
#
#$ -pe mpich 1
# pe request for MPICH. Set your number of processors here.
# Make sure you use the "mpich" parallel environment.
#
# Set to working directory to current directory
#$ -cwd
#
# Combine stderr and stdout to one file
#$ -j y
#
# Run job through bash shell
#$ -S /bin/bash
#
# Adjust MPICH procgroup to ensure smooth shutdown
export MPICH_PROCESS_GROUP=no
#
# Print useful information
echo "Got $NSLOTS slots."
echo "Program finish at: '/bin/date'"
```

To run "execute" with multiple model files, you must add the option "-run_on_sge" in the command line.

```
[Sompob@rocky POPPK]$/share/apps/PsN3.6.2/execute /home/Sompob/POPPK/test1.mod
test2.mod test3.mod test4.mod test5.mod -run_on_sge &
```

3.3.2 `bootstrap`

If you would like to do `bootstrap` in parallel with 24 CPUs, type

```
[Sompob@rocky POPPK]$/share/apps/PsN3.6.2/bootstrap -samples=1000 -seed=1234 -threads=24
/home/Sompob/POPPK/test-bootstrap.mod -dir=BS-test -run_on_sge &
```

`-threads` is the option to specify the number of CPUs. `-run_on_sge` `bootstrap` will generate and submit sub-jobs to the cluster automatically. This options must be used with `-threads` option.

3.3.3 `scm`

The following is the example of how to use `scm`.

```
[Sompob@rocky POPPK]$/share/apps/PsN3.6.2/bin/scm -config_file=/home/Sompob/POPPK/scm_config.scm
-threads=16 -dir=scm_test -run_on_sge &
```

3.3.4 Parallel NONMEM

You can run a NONMEM job in parallel by using `nmfe72gf` program. `nmfe72gf` can be copied from `/share/apps/X`.

This is the example of the job file for `nmfe72gf`.

```
#!/bin/bash
#$ -pe mpich 16
#$ -N nmparallel
#$ -cwd
#$ -m ae
#$ -j y
#$ -S /bin/bash

MPICH2_ROOT=/opt/mpich2/gnu/
export PATH=${MPICH2_ROOT}/bin:${PATH}
export NM72_INSTALL=/share/apps/nm72

$PWD/nmfe72gf $PWD/foce_parallel.ctl $PWD/foce_parallel.res
    -parafile=$PWD/mpilinux8.pnm [nodes]=$NSLOTS
```

3.4 R

The following is the example of the R codes and its job file for the cluster. You need to use the R option `--vanilla` in this case. See `man R` for more information.

```
# test1.R

x<-1
for(i in 1:10000000 ){
        x<-x+rnorm(1)
 }
write.table(x, "theanswer.txt")
```

The example of the job file for `test1.R`.

```
#!/bin/sh
#
# EXAMPLE MPICH SCRIPT FOR SGE
#
# Name of the job
#$ -N R-test
#
# pe request for MPICH. Set your number of processors here.
# Make sure you use the "mpich" parallel environment.
#$ -pe mpich 1
#
# Set to working directory to current directory
#$ -cwd
#
# Combine stderr and stdout to one file
#$ -j y
#
# Run job through bash shell
#$ -S /bin/bash
#
# Adjust MPICH procgroup to ensure smooth shutdown
export MPICH_PROCESS_GROUP=no
#
```

```
# Print useful information
echo "Program start at: `/bin/date`"
echo "Got $NSLOTS slots."

# Set machine list
cat $TMPDIR/machines

# Run program. Should use full path instead of relative path
/usr/bin/R --vanilla < /home/Sompob/TEST/R/test1.R

echo "Program finish at: `/bin/date`"
```

If you would like to run R in parallel, we have `Rmpi` and `doMPI` package installed already. This is an example of how to run R in parallel.

```
## testdoMPI.R ##
library(foreach,lib.loc="/share/apps/Rlib")
library(iterators,lib.loc="/share/apps/Rlib")
library(Rmpi,lib.loc="/share/apps/Rlib")
library(doMPI,lib.loc="/share/apps/Rlib")

cl<-startMPIcluster()
registerDoMPI(cl)

x<-seq(-8,8, by=0.5)
v<-foreach(y=x,.combine="cbind") %dopar% {
        r<-sqrt(x^2+y^2)+.Machine$double.eps
        sin(r)/r
}
persp(x,x,v)

closeCluster(cl)
mpi.quit()
```

And this is the example of the job file for `testdoMPI.R`.

```
#!/bin/bash
# Name of the job
#$ -N doMPI
#
# pe request for MPICH. Set your number of processors here.
# Make sure you use the "mpich" parallel environment.
#$ -pe mpi 16
#
# Set to working directory to current directory
#$ -cwd
#
# Combine stderr and stdout to one file
#$ -j y
#
# Run job through bash shell
#$ -S /bin/bash
#
# Adjust MPICH procgroup to ensure smooth shutdown
export MPICH_PROCESS_GROUP=no
#
# Print useful information
echo "Program start at: `/bin/date`."
echo "Got $NSLOTS slots."
```

```
# Run program. Should use full path instead of relative path
/opt/openmpi/bin/mpirun -np $NSLOTS /usr/bin/R --slave CMD BATCH testdoMPI.R

echo "Program finish at: `/bin/date`."
```

3.5 OpenBUGS

We have OpenBUGS on the cluster at `/share/apps/OpenBUGS312`.

```
#!/bin/sh
#
# EXAMPLE MPICH SCRIPT FOR SGE
#
# Name of the job
#$ -N BUGS-test
#
# pe request for MPICH. Set your number of processors here.
# Make sure you use the "mpich" parallel environment.
#$ -pe mpich 1
#
# Set to working directory to current directory
#$ -cwd
#
# Combine stderr and stdout to one file
#$ -j y
#
# Run job through bash shell
#$ -S /bin/bash
#
# Adjust MPICH procgroup to ensure smooth shutdown
export MPICH_PROCESS_GROUP=no
#
# Print useful information
echo "Program start at: `/bin/date`"
echo "Got $NSLOTS slots."
# Set machine list
cat $TMPDIR/machines

# Run program. Should use full path instead of relative path
/share/apps/OpenBUGS312/bin/OpenBUGS < /home/Sompob/TEST/OpenBUGS/test1.txt

echo "Program finish at: `/bin/date`"
```

3.6 PhyML

PhyML(Phylogenetic estimation using Maximum Likelihood) is at `/share/apps/PhyML`.
Here is the example of a job file for PhyML (parallel).

```
#!/bin/sh
#$ -pe mpich 8
#$ -cwd
#$ -j y
#$ -S /bin/bash

export MPICH_PROCESS_GROUP=no
echo "Program start at:`/bin/date`"
echo "Got $NSLOTS slots."
echo "Machines:"
```

```
cat $TMPDIR/machines

# for parallel PhyML
/opt/openmpi/bin/mpirun -n $NSLOTS  /share/apps/PhyML/phyml-mpi YOURINPUT...
```

The example files present in this document can be copied from `/home/Sompob/examples/` and `/home/Sompob/TEST/`.

If you have any comments, suggestions or questions, please contact sompob@tropmedres.ac.

References

- Quick Linux Tutorial, *Jiří Vogel*, `http://www.fsid.cvut.cz/cz/U201/LINUX.HTML`
- Rocks Clusters, `http://www.rocksclusters.org/`
- Parallel Programming with MPI, *Peter S. Pacheco*, `http://www.cs.usfca.edu/~peter/ppmpi/`
- NAMD - Scalable Molecular Dynamics, `http://www.ks.uiuc.edu/Research/namd/`
- Perl-speaks-NONMEM, `http://psn.sourceforge.net/`
- OpenBUGS, `http://www.openbugs.info`
- R, `http://cran.r-project.org/`
- Numerical Recipes, `http://www.nr.com/`
- Mathematica, `http://www.wolfram.com/`