Logging in to the CRAY

Cray Hostname: cray2.colostate.edu
Cray IP address: 129.82.103.183

On a Mac

1. Open Terminal
2. type **ssh username@cray2.colostate.edu** where username is your account name
3. enter password
4. to change password type **passwd2** then **enter**

On a PC

1. Open Putty
2. ssh to cray2.colostate.edu with your username
3. enter password

Basic Unix commands for the Cray

Moving around in Unix

- **ls** $\rightarrow$ gives the contents of the directory of interest
- **cd path** $\rightarrow$ changes directory to path
- **cd..** $\rightarrow$ takes you one level backwards in the directory structure
- **cd** (or **cd**)$\rightarrow$ takes you to your home directory
- **pwd** $\rightarrow$ gives the filepath of the curent directory
- **name*** $\rightarrow$ wildcard character, finds all files in the current directory that start with name...
- ***name** $\rightarrow$ wildcard character, finds all files in the current directory that end with name...
- ***name*** $\rightarrow$ wildcard character, finds all files in the current directory that contain name...
- **logout** $\rightarrow$ logout of the Cray

Basic Unix commands for the CRAY

Copying, Moving, and Removing Files in Unix

- **cp file1 file2** $\rightarrow$ copies the file named file1 to the file named file2
- **cp file1 /path/file2** $\rightarrow$ copies the file named file1 to the file named file2 in the directory file/path
- **mv file1 file2** $\rightarrow$ renames the file named file1 to file2 (note, this removes file1)
- **mv file1 /path/file2** $\rightarrow$ moves the file named file1 to the file named file2 in the directory file/path
- **rm filename** $\rightarrow$ permanently deletes the file filename
- **mkdir directoryname** $\rightarrow$ makes the directory named directoryname
- **rmdir directoryname** $\rightarrow$ removes the empty directory named directoryname
- **rm -r directoryname** $\rightarrow$ removes the directory named directoryname and all the files in directoryname

Text Editing - Emacs

There are two primay Unix editors, vi and Emacs. I know Emacs and will focus on this. There are also many Emacs ports that include links to R, Latex, Sweave, Knitr, and other tools to improve workflow.

In Unix

- **emacs** $\rightarrow$ opens the text editor emacs
- **emacs filename** $\rightarrow$ opens the file filename in emacs

In Emacs

- There are two primary command keys in Emacs - Command(C-) and Meta(M-)
- **C-** is the **CTRL** key on a Mac
- **M-** is the **ESC** (or **ALT**) key on a Mac

Emacs Commands

- **C-x C-c (CTRL-x CTRL-c)** $\rightarrow$ closes the emacs document and asks if you want to save if there are unsaved changes
- **C-x C-s** $\rightarrow$ saves the current emacs document
- **C-v** $\rightarrow$ page-down one screen
- **M-v (ESC-v or ALT-v)** $\rightarrow$ page-up one screen
- **C-a** $\rightarrow$ goes to the beginning of the current line
- **C-e** $\rightarrow$ goes to the end of the current line
- **C-k** $\rightarrow$ deletes the current line
- **C-y** $\rightarrow$ restores the deleted line

Editing your .bash_profile - To setup the Cray for each login and link the R program to the home directory

When logged into your home directory on the Cray, type

emacs .bash_profile

In Emacs enter the text:

```
export PATH=/apps/R-2.14.2/bin:$PATH
export LD_LIBRARY_PATH=/opt/gcc/4.1.2/cnos/lib64:/opt/gcc/4.4.4/snos/lib64/:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/apps/openmpi-1.6.2/lib/:$LD_LIBRARY_PATH
export TMP=$HOME/lustrefs/tmp/
export MPI_LIBPATH=/apps/openmpi-1.6.2/lib
export MPI_INCLUDE_PATH=/apps/openmpi-1.6.2/include/
export RMPI_TYPE=''OPENMPI''
```

then enter **C-x C-s (CTRL-x CTRL-s)** to save and **C-x C-c** to exit emacs. If you copy and paste this, Type logout to exit the Cray system and ssh back into the Cray to start a new session with the .bash_profile. Test that this has been successful by typing **R** in your home directory.

Transferring files to and from the CRAY

Windows

- use winSCP → this is a gui interface and self explanatory

Mac

- **sftp username@cray2.colostate.edu** → log in to the Cray using sftp
- **put /path1/filename1 /path2/filename2** → uploads filename1 from /path1/ on the local system to /path2/filename2 to the Cray
- **get /path1/filename1 /path2/filename2** → downloads filename1 from /path1/ on the Cray to /path2/filename2 on the local system
- **quit** → logout of sftp

Setting up the Cray for R version 2.14.2

Copy **RMPISNOW** into your **lustrefs** directory by typing

cp /apps/R-2.14.2/lib64/R/library/snow/RMPISNOW /home/username/lustrefs/RMPISNOW where **username** is your username and there is a space between **cp /apps** and between **RMPISNOW /home/**. If you are unsure about your username type **whoami**.

Create a R temporary directory **tmp** under **lustrefs** by typing **mkdir tmp** while in the **lustrefs** directory. Then type **export TMP=$HOME/lustrefs/tmp/**

Useful Cray commands

- **qsub filename** $\rightarrow$ submits the job filename to the queue using batch commands
- **qstat** $\rightarrow$ shows the status of jobs in queues
- **qstat -Q** $\rightarrow$ shows all available queues (brief)
- **qstat -Qf** $\rightarrow$ shows all available queues (full)
- **qstat -u username** $\rightarrow$ shows the status of jobs that belong to you
- **xtnodestat** $\rightarrow$ shows the status of compute nodes
- **qdel jobid** $\rightarrow$ deletes the job with job ID = jobid from the batch queues as long as you are the owner of the job

Batch Script - to submit parallel jobs

To create a batch script, type **emacs** and then enter:

```
#!/bin/bash
#PBS -N job.name
#PBS -j oe
#PBS -l mppwidth=24
#PBS -l walltime=01:00:00
#PBS -q small
cd $PBS_O_WORKDIR
date
aprun -n 24 RMPISNOW <script.R> out.txt
date
```

and then type **C-x C-s (CTRL-x CTRL-s)** and save the file as **run.txt** then type **C-x C-c** to close emacs.

Batch Script description

- **#!/bin/bash** specifies the Unix shell environment for the batch job
- **-N jobname** renames the output file as jobname
- **-j oe** specifies that output and errors will be combined
- **-l mppwidth** specifies the number of cores to allocate to your job. The number of cores should be a multiple of 24 and not exceed 1,344
- **-l walltime** specifies the maximum amount of time in hours, minutes and seconds that the job may run
- **-q small** specifies which queue the job should be run in. There are four options: **small**, **medium**, **large**, and **ccm_queue**
- **PBS_O_WORKDIR** contains the absolute path to the directory from which you submitted your job

Parallel Code Example - using the **snowfall**, **parallel**, and **rlecuyer** libraries

Let's try some parallel code on our desktop/laptop machines and see how this works. See the code **parallel.example.R**

Any **for** loop where each iteration of the loop is independent of the other iterations (no MCMC) can be written as an apply command (**apply, sapply,** and **lapply**) and this can be parallelized by using **sfApply, sfSapply,** and **sfLapply** inside the **snowfall** library.

Parallel Code Example - using the **snowfall**, **parallel**, and **rlecuyer** libraries

- Detect the number of cores on your system using **cps ¡- detectCores()**
- Setup the **snowfall** cluster using **sfInit(parallel = TRUE, cpus = cps**
- Make sure to set up the cluster random number generator using **sfClusterSetupRNG()**
- Make sure to export the functions, variables, and libraries needed for parallel calculation by using **sfExport('function')**, **sfExport('variable')**, **sfExportAll()**, and **sfLibrary('library')**.
- End the snowfall cluster using **sfStop()**

Parallel Code Example - using the **snowfall**, **parallel**, and **rlecuyer** libraries

- Detect the number of cores on your system using **cps ¡- detectCores()**
- Setup the **snowfall** cluster using **sfInit(parallel = TRUE, cpus = cps**
- Make sure to set up the cluster random number generator using **sfClusterSetupRNG()**
- Make sure to export the functions, variables, and libraries needed for parallel calculation by using **sfExport('function')**, **sfExport('variable')**, **sfExportAll()**, and **sfLibrary('library')**.
- End the snowfall cluster using **sfStop()**

Let's put it all together - running code on the Cray

- **sftp** the files **script.R** and **run.txt** onto your **lustrefs** directory on the Cray
- Submit the job to the Cray compute nodes by typing **qsub run.txt**
- Check the status of the job on the Cray by typing **qstat**
- Review the files in your **lustrefs** directory by typing **ls**
- Once the job is finished check the results by typing **emacs out.txt** to see the R code and **emacs my.first.run.oxxxxxx** to see the Cray outpur and errors, where **xxxxxx** is the job id number

How to parallelize your own code for the Cray

- Setup the cluster **cl** $\leftarrow$ **makeCluster()**
- Export the cluster using **clusterExport(cl, ls())**
- Setup the cluster random number generator using **clusterSetupRNG(cl)**
- Can parallelize your code using **parApply**, **parSapply**, and **parLapply**
- Stop the cluster with **clusterStop()**

Some notes about the Cray

- There are 24 cores on each node of the Cray, make sure you don't waste these resources by not using all the cores
- It appears that you can have 23 slaves with one master and the master sits idle (at least from my experience)
- Small jobs have higher priority
- Jobs with fewer numbers of cores needed are easier to schedule (and get run quicker)
- Make sure you have enough runtime to complete the job otherwise it will be cut-off
- Small queue - less than 1 hour runtime, highest priority, can have up to 20 jobs at once
- Medium queue - less than 24 hour runtime, medium priority, can have up to two jobs at once
- Large queue - less than 7 days runtime, lowest priority, can have only one job at a time