Simple Parallel Computing in R

Libo Sun

libosun@rams.colostate.edu

Department of Statistics
Colorado State University

October 15, 2014
Outline

1. What is Parallel Computing in R and Why?
2. Parallel Computing in R on multi-core computers.
3. What is the Cray?
4. Parallel Computing in R on the Cray.
What is Parallel Computing in R and Why?

- Many statistical analysis tasks are computationally very intensive.

- Often multiple cores are available. However, R only uses a single core.

- Many problems are “embarrassingly parallel”.
  - Split the problem into many smaller parallel tasks for computing simultaneously.
  - Usually no dependency between the parallel tasks.

- **A rule of thumb**: if you can wrap your task in an `apply` function or one of its variants.
Ideal Performance Improvement

- $p$ cores should be $p$ times faster than one core.

<table>
<thead>
<tr>
<th>One core</th>
<th>60 cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 minute</td>
<td>1 second</td>
</tr>
<tr>
<td>1 hour</td>
<td>1 minute</td>
</tr>
<tr>
<td>1 day</td>
<td>30 minutes</td>
</tr>
<tr>
<td>1 week</td>
<td>3 hours</td>
</tr>
<tr>
<td>1 month</td>
<td>12 hours</td>
</tr>
</tbody>
</table>
Mater/Slave parallel model

- **Ideal:**

Single process:

Master:

Slave 1:

Slave 2:

Slave 3:

Slave 4:

Slave 5:

- **Realistic:**

Master:

Slave 1:

Slave 2:

Slave 3:

Slave 4:

Slave 5:
If you are using Mac or Linux, congratulations! 
**multicore** (or **parallel** in R > 2.14.0) is surprisingly easy!

- Substitute the `lapply` function with `mclapply`.

Bad news: Both **multicore** and **parallel** don’t support Windows!

- Use **snow** or **snowfall**. (Talk later)
A simple example of parallel

```r
> library(parallel)
> detectCores()
[1] 2
> Iteration <- function( iter=1, n=100 ){
+   x <- rnorm( n, mean=2, sd=2 )
+   eps <- runif( n, -3, 3 )
+   y <- 1 + 2*x + eps
+   fit <- lm( y ~ x )
+   return( cbind( fit$coef, confint( fit ) ) )
> }

> nsim <- 10000

> system.time(lapply(1:nsim, Iteration, n=100))
user  system elapsed
25.712  0.224  25.960

> system.time(mclapply(1:nsim, Iteration, n=100))
user  system elapsed
13.924  0.185  15.214
```
ISTeC Cray High Performance Computing System at Colorado State University.

The ISTeC Cray is a XT6m model with 1,248 cores (computing devices), 1.6 terabytes of main memory (about 13 trillion bits) and 32 terabytes of disk storage.

12 interactive compute nodes (288 cores) for testing, developing, and debugging and 40 batch compute nodes (960 cores) for large jobs.

Only a single job can be run at a time on any node, consisting of 24 cores. (Do not waste)
Cray System Architecture

Cray System Architecture

Login Nodes
- User shell
  - Access files
  - Submit jobs

Compute Nodes
- Application
  - Standard I/O layer
  - Lustre

System Interconnect
- OST
- OST
- OST
- OST
- MDT

OSS Node

MDS Node
Preparation

- Apply an account at ISTeC Cray website.

- To access the Cray: Use SSH (like PuTTy) and SFTP or SCP (like WinSCP). Check Cray’s User’s Guide for detail.

- R 2.14.2 is installed on the Cray under /apps directory. (Use “ls /apps” to check)

- Access R by entering “/apps/R-2.14.2/bin/R” (no quotes).

- Create a R temporary directory “tmp” under “lustrefs” by entering “mkdir tmp”. Then enter “export TMP=$HOME/lustrefs/tmp/”.
Preparation

- To save typing this all the time you can place “export PATH=/apps/R-2.14.2/bin:$PATH” and “export TMP=$HOME/lustrefs/tmp/” in a “.bash_profile” file (no quotes) in your home directory by typing “vi .bash_profile”.

- Also you need to place “export LD_LIBRARY_PATH=/opt/gcc/4.1.2/cnos/lib64:/opt/gcc/4.4.4/snos/lib64/:$LD_LIBRARY_PATH” in the “.bash_profile” file.

- Enter “:wq” to save and exit.

- Then just enter “R” to launch R on the login node.
Preparation

Simple Parallel Computing in R

Libo Sun

What and Why?

Multi-core Computers

What is the Cray?

Parallel Computing in R on the Cray.

Summary

References
Simple Parallel Computing in R

Libo Sun

What and Why?

Multi-core Computers

What is the Cray?

Parallel Computing in R on the Cray.

Summary

References

---

Enter `library()` to check all libraries on the Cray.

Do **NOT** run your code on the login node! It just likes your personal computer. (Only has two cores)
The R package **snow** (Simple Network of Workstations)

- A master R process, running either interactively or as a batch process, creates a cluster of slave R processes that perform computations on behalf of the master.
- Communication between master and slaves
  - Socket interface
  - MPI (Message-Passing Interface) via **Rmpi** package.
  - PVM (Parallel Virtual Machine) via **rpvm** package.
  - NWS (NetWorkSpaces) via **nws** package.
- For multi-core computers, the simplest choice is socket.
- Use MPI via **Rmpi** package on the Cray.
The R package **snow** (Simple Network of Workstations)

**Basic functions:**
- **makeCluster** initializes a cluster.
- **clusterExport** exports objects to each slave.
- **clusterEvalQ** can load required packages on all slaves.
- **clusterSetupRNG** sets up random number generation. It ensures slaves produce independent sequences of random numbers.
- **parLapply, parSapply, and parApply** are parallel versions of `lapply`, `sapply`, and `apply`.
- **stopCluster** stops the cluster.
A simple example of snow on multi-core computer

```r
> library(snow)
> cl<-makeCluster(2,type='SOCK')  #Start a socket cluster of 2 R slaves
> #Random number generation, need 'rlecuyer' package
> clusterSetupRNG(cl)
Loading required package: rlecuyer
[1] "RNGstream"
> clusterExport(cl,ls())  #Export everything to each slave
> system.time(lapply( 1:nsim, Iteration, n=100))
   user  system elapsed
 26.13   0.03  26.35
> system.time(parSapply(cl, 1:nsim, Iteration, n=100))
   user  system elapsed
 0.08   0.01  15.54
> stopCluster(cl)  # Stop the cluster
```
Submit the job to compute nodes on the Cray

Interactive compute nodes:
- Use "aprun -n 24 RMPISNOW <Rcode.R >output.txt"
- "aprun -n 24 RMPISNOW" starts a MPI cluster of 23 R slaves and one master on the Cray.
- Copy "RMPISNOW" to the directory from which you want submit your job by entering "cp /apps/R-2.14.2/lib64/R/library/snow/RMPISNOW ."

Batch compute nodes:
- Torque/Moab/PBS batch queuing system for managing batch jobs.
- Must create a text file (batch script) that contains Torque/PBS commands.
- "qsub filename" to submit the batch job.
A sample batch script

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

- `-q small` specifies the "small" batch queue.
- `-l mppwidth` and `-n` should be the same.
- Need "RMPISNOW" file as well.
- "jobname.o1234" would be created when job is done, where "1234" is the job ID. It contains both standard output and standard error from the Cray.
## Batch queues

<table>
<thead>
<tr>
<th>Queue</th>
<th>Priority</th>
<th>Walltime</th>
<th>Max num of jobs per user</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>high</td>
<td>1 hr.</td>
<td>20</td>
</tr>
<tr>
<td>medium</td>
<td>medium</td>
<td>24 hrs.</td>
<td>2</td>
</tr>
<tr>
<td>large</td>
<td>low</td>
<td>1 week</td>
<td>1</td>
</tr>
</tbody>
</table>
> # obtain a MPI cluster of 23 R slaves started with 'aprun'
> cl<-makeCluster()
>
> # Random number generation, need 'rlecuyer' package
> clusterSetupRNG(cl)
> [1] "RNGstream"
>
> # Export everything to each slave
> clusterExport(cl,ls())
>
> system.time(lapply( 1:nsim, Iteration, n=100))
> user  system elapsed
> 35.850  0.004  35.867
>
> system.time(parSapply(cl, 1:nsim, Iteration, n=100))
> user  system elapsed
> 1.896  0.000  1.897
>
> # Stop the cluster
> stopCluster(cl)
Comments

- Communication is much slower than computation.
- Use shorter “walltime” to have higher priority.
- Be mindful of the shared resources.
- The number of cores should be a multiple of 24.
- Go into the “lustrefs” directory for all parallel jobs.
- **snowfall** was built as an extended abstraction layer above the **snow**. It has some advantages over **snow**:  
  - Better error handling.
  - More functions for common tasks in parallel computing.
  - All functions work in sequential execution.
  - Bad news: Need some adjustments in “RMPISNOW” file for using **snowfall** on the Cray.
Some useful commands on the Cray

- "ls" lists the contents of a directory.
- "mkdir new" creates a "new" directory.
- "cp file1 file2" copies file1 to file2.
- "rm file" removes the "file". (Careful, no trash can)
- "cd new" changes to "new" directory.
- "cd.." goes back one directory.
- "qstat" shows the status of jobs in all queues.
- "xtnodestat" shows the status of compute nodes.
- "qdel jobid" deletes the job with job ID = jobid from the batch queues.
The status of compute nodes

Legend:
- nonexistent node
- service node
- free interactive compute node
- free batch compute node
- allocated, but idle compute node
- suspect compute node
- down compute node
- Y down or admindown service node
- Z admindown compute node

Available compute nodes: 11 interactive, 23 batch

<table>
<thead>
<tr>
<th>Job ID</th>
<th>User</th>
<th>Size</th>
<th>Age</th>
<th>command line</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>pramit</td>
<td>1</td>
<td>0h49m</td>
<td>a.out</td>
</tr>
<tr>
<td>b</td>
<td>jhung</td>
<td>3</td>
<td>1h54m</td>
<td>q3d</td>
</tr>
<tr>
<td>c</td>
<td>npolley</td>
<td>1</td>
<td>55h34m</td>
<td>gtxsolve</td>
</tr>
<tr>
<td>d</td>
<td>npolley</td>
<td>2</td>
<td>55h34m</td>
<td>sleep</td>
</tr>
<tr>
<td>e</td>
<td>hku</td>
<td>2</td>
<td>0h00m</td>
<td>sun</td>
</tr>
<tr>
<td>f</td>
<td>zheikes</td>
<td>2</td>
<td>44h29m</td>
<td>2Grd</td>
</tr>
<tr>
<td>g</td>
<td>hku</td>
<td>2</td>
<td>0h00m</td>
<td>sun</td>
</tr>
<tr>
<td>h</td>
<td>jhung</td>
<td>3</td>
<td>0h00m</td>
<td>q3d</td>
</tr>
<tr>
<td>i</td>
<td>hku</td>
<td>2</td>
<td>0h00m</td>
<td>sun</td>
</tr>
</tbody>
</table>
Summary

To do Parallel Computing in R on the Cray:

- One time work (after you log in):
  - Create “.bash_profile” for R location and temporary directory in your **home directory**.
  - Copy “RMPISNOW” from **snow** library to where you want to work at.

- Interactive nodes: “aprun -n 24 RMPISNOW <Rcode.R >output.txt”

- Batch nodes: Create a batch script and use “qsub filename” to submit.
http://www.stat.uiowa.edu/ luke/R/cluster/cluster.html


http://www.sfu.ca/ sblay/R/snow.html

http://cran.r-project.org/web/views/HighPerformanceComputing.html


http://www.imbi.uni-freiburg.de/parallel/