Parallel with 'snow'
An Example

Feng Yi

April 27, 2012

Resources

Slides can be downloaded from:

http://www.stat.umn.edu/∼yixxx064/parallel/snowExample.pdf

Problem Setting

Suppose you have covariance matrix Σ, with

$$p = 500, n = 5000$$

You want to do simulation: generate 100 replicates of data, estimate precision matrix Σ^{-1}. Finally, you want to summarize the 100 estimates.
Estimation error is defined as:

$$error = ||\hat{\Sigma}^{-1} - \Sigma^{-1}|| \quad (1)$$

NonParallel Version of R Code

```
set.seed(12345)
for (wp.i in 1:n.trials){
    error.RD <- rep(0, 100)
    ### Step 1: generate data ###
    data <- rmvnorm(n, sigma = covMatrix)
    ### Step 2: calculation ###
    inv.mat.est = solve(cov(data))
    ### Step 3: save your results ###
    error.RD[wp.i] = norm(inv.mat.est - ar1, type="1")
}
```

The 'n.trials' is total number of jobs.

Parallel Version of R Code: 1

Write up a wrapper function:

```
wrapper.function <- function(n.trials) {
    error.RD <- NULL
    for (wp.i in 1:n.trials) {
        data <- rmvnorm(n, sigma = covMatrix)
        inv.mat.est = solve(cov(data))
        error.RD[[wp.i]] = norm(inv.mat.est - ar1, type="1")
    }
    list(error.RD = error.RD)
}
```

The 'n.trials' is the number of jobs on each node.

Parallel Version of R Code: 2

```
library(snow)
library(rlecuyer)
### start up a cluster ###
cl <- makeCluster(5, type="MPI")
### Get all ‘‘library’’s into each core ###
invisible(clusterEvalQ(cl, {library(mvtnorm); NULL}))
### Set random seed(s) ###
clusterSetupRNG(cl, type="RNGstream", seed=10)
### Get all parameters, and functions into each core ###
clusterExport(cl, list("ar1", "covMatrix", "n"))
### Apply parallel function ###
parRet <- clusterApply(cl, rep(100/5, 5), wrapper.function)
### Stop cluster ###
stopCluster(cl)
```

Useful Suggestions

- Save one node for your parent job.
- Be careful about the memory restriction.
- Use some special index for the "for" loop of wrapper function.
- The children nodes have the same priority as your parent node in the example.

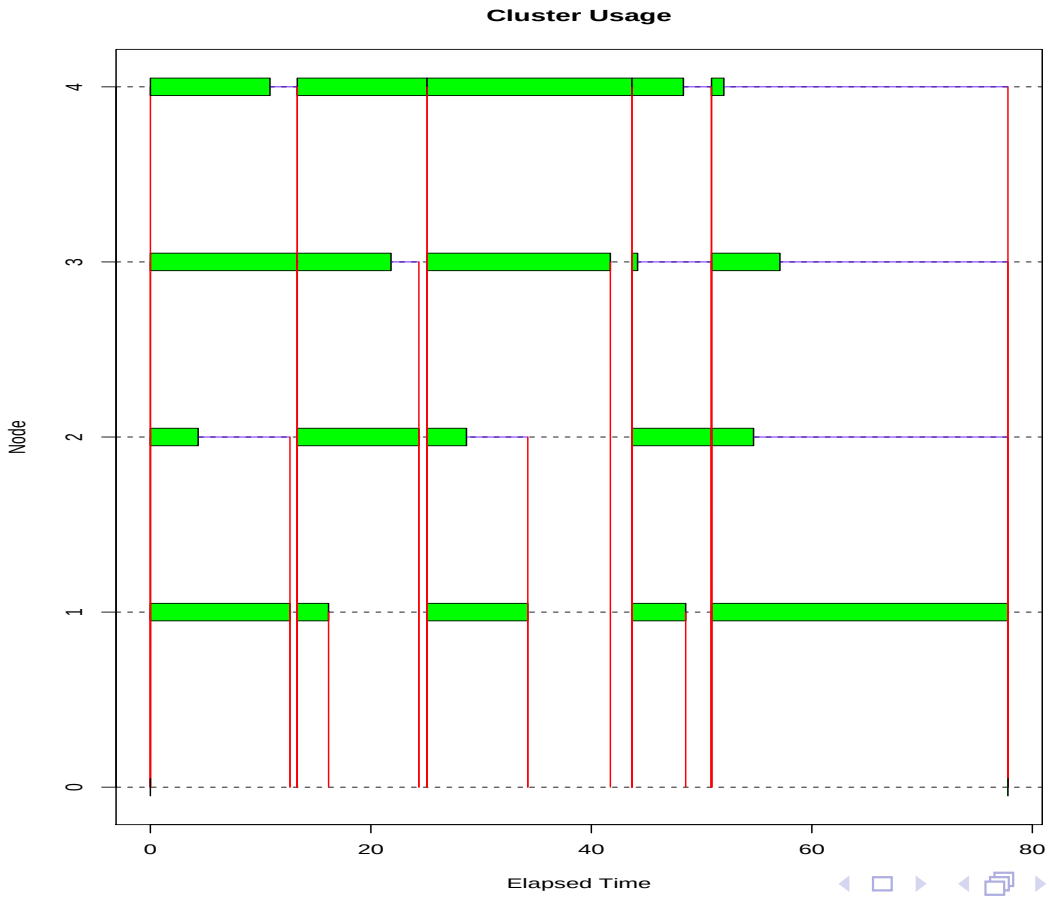
Load Balancing

Suppose you have 'n' jobs to 'clusterApply' on 'm' nodes with $n > m$. The 'm' nodes start 'm' jobs simultaneously, and they will not start next jobs until all the previous 'm' jobs are done.

Load balancing becomes important when it takes quite different time to finish each of the 'n' jobs, and $n > m$. You hope every node starts the next job immediately after it finishes the previous one. If this is the case, you realize load balancing.

Load Balancing Problem with 'clusterApply'

```
> sleeptime <- abs(rnorm(20, 5, 10))
> tm <- snow.time(clusterApply(cl, sleeptime, Sys.sleep))
```



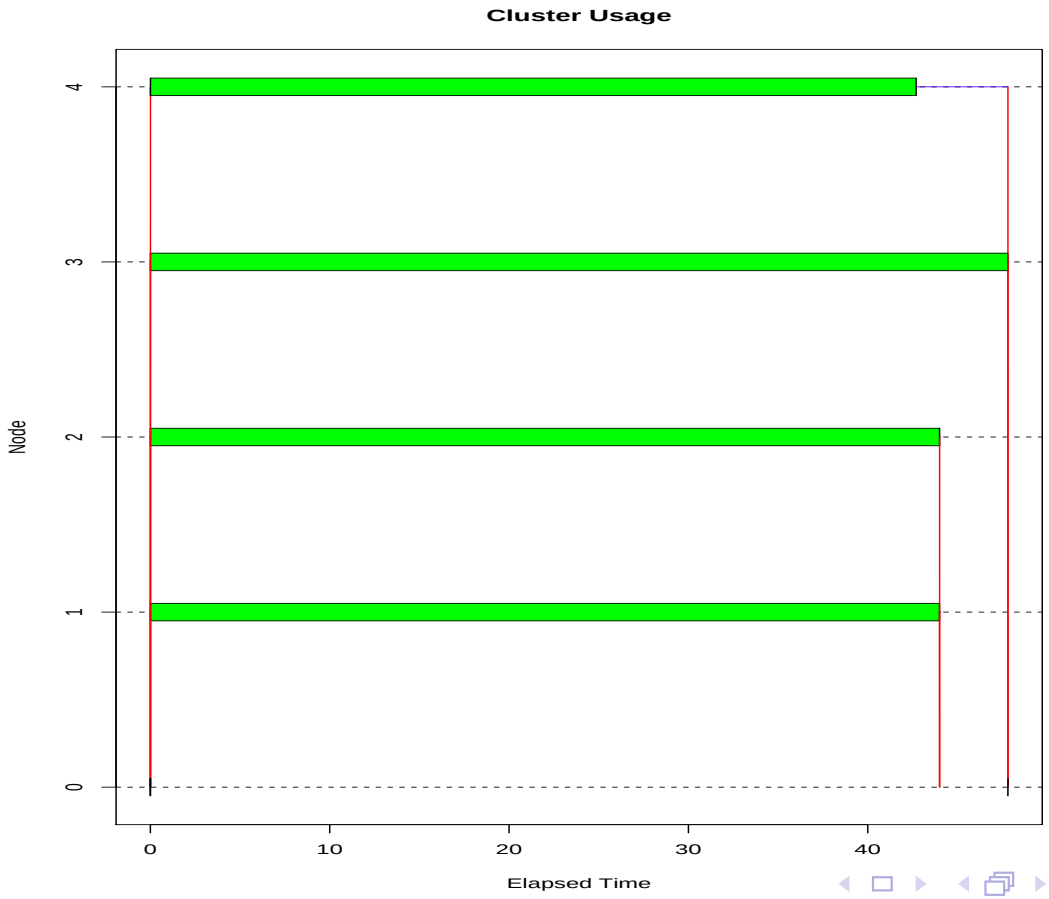
'parLapply' Function

"parLapply" :

- a high-level snow function
- it realizes task chunking
- it partially solves loading balance problems
- it will have less I/O operations when the number of jobs are greater than the number of nodes
- most useful snow function

Load Balancing with 'parLapply'

```
> sleeptime <- abs(rnorm(20, 5, 10))
> tm <- snow.time(parLapply(cl, sleeptime, Sys.sleep))
```



'parLapply' Example

After you start up a cluster, set random seeds, get useful library, function into each node, you can try:

```
### The length of ''flag'' is the total number of jobs. ###
wrapper.fun2 <- function(flag, n, covMatrix, ar1) {
    ### Step 1: generate data ###
    data <- rmvnorm(n, sigma = covMatrix)
    ### Step 2: calculation ###
    inv.mat.est = solve(cov(data))
    ### Step 3: save your results ###
    error = norm(inv.mat.est - ar1, type="1")
}
Error = unlist(parLapply(cl, rep(1,n.trials), wrapper.fun2,
                n=n, covMatrix=covMatrix, ar1=ar1))
```

'parLapply' returns a list of 'wrapper' function return value.

Reference

- Parallel R - Data Analysis in the Distributed World, *by Q.Ethan McCallum, Stephen Weston*
- Parallel Processing here at the School of Statistics, *by Charles J. Geyer*