Parallel computing in R

Robert Bagchi

March 17, 2016

1 Introduction

Many tasks we undertake in statistical computing involve a lot of repitition. Often each repeat is independent of the others - for example, we might want to make independent random draws from a distribution. These tasks are often described as "embarrassingly parallel" because we do not have to do them in sequence - they could all be done at the same time so a job that took (number of repeats) x (time of each repeat) could be reduced to time of each repeat, should we have sufficient resources. Most modern computers are capable of at least 4 parallel processes, which means you could accomplish tasks almost 4 times as fast!

Below is a simple example of a repetitive task.

```
> # let us do a simple calculation involving a lot
> ## of repetition, each task of which takes some time
>
> ## one common task is inverting a matrix - you
> ## have unconsciously done this many times when fitting
> ## linear models. Let us build a huge matrix (1000 x 1000)
> ## and invert it
> m <- matrix(runif(1e6), ncol=1e3, nrow=1e3)
> test <- solve(m) ## this takes some time
> ## to find out how long it takes, let us time it
> ## with system time
>
> system.time(test <- solve(m)) ## about 0.9 seconds

   user  system elapsed
   0.94    0.00    0.94

> ## now let us do this 20 times in sequence.
> ## I will use a for loop first.
>
> result <- list()
> ## this takes about 20 x the time
```

```
> system.time(for(i in 1:20){
+   m <- matrix(runif(1e6), ncol=1e3, nrow=1e3)
+   result[[i]] <- solve(m)
+ })

   user  system elapsed
  20.78    0.20   21.15

> ## we can sometimes speed up a little by using the apply
> ## functions.
> invertmatrix <- function(){
+   m <- matrix(runif(1e6), ncol=1e3, nrow=1e3)
+   return(solve(m))
+ }
> ## First write the function we want to repeat
> system.time(result <- sapply(1:20, function(i){
+   invertmatrix()
+ }, simplify=FALSE))

   user  system elapsed
  20.75    0.15   20.99

>
> ## but doesn't really make a difference here
```

2 running parallel tasks on your computer

Parallel support is provided by the parallel package, which ships with base R. It includes a number of functions that mirror the apply series of functions, for example, sapply and lapply. The parallel versions of these functions are parSapply and parLapply.

```
> library(parallel)
> ## first set up  a cluster - asking for 3 cpus
> cl <- makeCluster(spec = 3)
> ## we then have to pass each of those nodes the function
> clusterExport(cl, varlist = 'invertmatrix')
> system.time(result <- parSapply(cl, 1:20, function(i){
+   invertmatrix()
+ }, simplify=FALSE))

   user  system elapsed
   0.23    0.56   12.60

> ## and it should complete the task in less time
>
> ## after you're done you should close the cluster
> stopCluster(cl)
```

The reason it's not 1/3 rd the time is there is some overhead when using parallel processing - the computer needs to pass data among different parts of the memory and so on. Still, if you are doing this on a machine with 10s of cores, you get quite substantial speedup!