Classroom Demonstrations of Parallel Processing for Computational Statistics

Prof. Eric A. Suess
California State University, East Bay

JSM 2013

Abstract:

With the common availability of multicore and multiprocessor computers, students of Statistics have the ability to run statistical computations in parallel. Within R there are various packages to implement parallel processing and there are some packages that have built-in options to run in parallel.

We introduce parallel processing for the mean and standard deviation. These examples show the idea of dividing the overall computation into separate calculations that can be done in parallel. Other examples include, running linear regression calculations in parallel, using the option in the bootstrap command to run parallel bootstrap calculations, and to run clustering in parallel.

To show the benefits parallel processing we examine the time taken to perform the calculations. Not all statistical calculations benefit from parallel processing, but many do.

Introduction:

We became interested in the use of parallel computation (distributed computing) and to a lesser extent in the use of parallel data storage (distributed data storage) during the excitement of the NetFlix competition that was offered a few years ago (2009). Learning that the winners of the competition used amazon's EC2 cluster and S4 storage inspired us to investiage and learn about these new computing environments, now referred to as "cloud computing". The Heritage Health Data competition was also very exciting (2012). And now with kaggle, being a location on the Internet for these types of competitions, there are current open data analysis competitions posted regularly. Many of the kaggle completions focus on Large Data problems which benefit from the use of parallel computation.

These kinds of computing techniques and facilities currently seem far from the introductory course work in Statistics at all levels, lower division and upper division Statistics courses, and first year graduate courses in Statistics. However, with some effort to motivate the idea of parallel computing at earlier stages in Statistics Education would be very helpful, then covering practical applications of parallel computation earlier in the undergraduate curriculum and Master's level curriculum could be very exciting to students.

Why is parallel computation not being discussed earlier in Statistics?

This is an important question! My opinion is that because the data problems introduced in Statistics courses at all levels usually include small and moderate sized data sets, therefore the use of parallel computing is mostly counter productive. While this is true, the opportunity to interduce some of the fundamental computing ideas is missed and the opportunity to engage students early about a direct connection to parallel computing, that is very common in the student's daily life, is missed. It also misses the opportunity to build a strong foundation in computing earlier in Statistics majors. Finally, it also misses an excellent opportunity for Statistics students to collect real data about there own computing. All of these issues relate to the larger emerging field of Analytics.

In the end, ...

Students of Statistics need to become much more capable and knowledgeable users of their own computers, which are now all inherently capable of parallel computation (Core2Duo, iCore3, iCore5, iCore7, 2, 4, 8 cores respectively, for example). Statistics educators should try to incorporate more use of these advanced computing devices in a way that better reflects the advanced applications for data analysis that have developed recently.

What is parallel computation and distributed computing?

The idea of parallel computing is that parts of an overall calculation can take place simultaneously, hence in parallel.

A student might ask, “What is the multicore processor in my computer useful for?”

The answer in a Statistics class hopefully will be, “For computing with data.” This answer could be followed by a simple example of computing a sum (or mean) in parallel.

Parallel processing in R

Most of the data sets used in Statistics are small and most of the software used is not natively able to parallel process. The clear benefit is not there at the early levels of statistics.

R does not have native parallel processing built-in. However, like everything in R, there is a library (I should say libraries) for parallel computing. In R it has become much easier to introduce the ideas of parallel computation with the **parallel** and also **multicore** libraries.

Others are **snow**, **snowfall**, **SPRINT**, etc.

R for and apply()

To perform parallel computations one needs to understand the basic **for** loop. This is the most basic computer programming structure to repeat a calculation a fixed number of times. In R the **for** loop is easily implemented. However, the **for** loop is not efficient in R and the **apply()** function is usually recommended for use if possible. There are many variations on the **apply()** function available in R.

The **apply** function applies a function to different parts of an overall structure. For example, the **sum()** function can be used with the **apply()** function to compute the row or column totals of a matrix. In the various libraries in R for parallel computation there are variations on the **apply()** function, such as **parApply()**.

What is a cluster and how to start a cluster on a multicore computer?

I am not an expert in clusters or super computing, but here goes, to the best of my knowledge. The idea of a cluster is that of a collection of usually identical computers connected together for the purpose of running computers programs so that parts of the computations can be done simultaneously.

Computers connected together allow for what is called parallel processing. In the past computer chips only had single processing capability, so connecting two or more single processing computers allowed for parallel processing. The advantage with parallel processing is that more computation can be done at the same time, leading to a time savings. This should lead to more efficient computing, however communication times are much slower over network connections, which are used to connect the computers, that internal connections within a single computer.

Parallel processing is more effective for larger computing problems. There are potentially larger time improvements that can be achieved in these situations. Parallel processing can lead to less efficiencies in smaller computational problems, in these situations the communication times may increase the overall time to complete the computations. Hence, there is a need for monitoring the system times to make sure using a cluster of computers is worthwhile.

Currently multicore computers are common. With the slowdown of the advancement of processor speed improvements, computer chips are now designed with multiple processor cores on single chips. This effectively gives a multicore processor the ability to run multiple computing tasks at the same time. So a multicore computer has the capability of parallel processing within a single computer without the need for network connected computers in a cluster. It is this setting that we approach the issue of parallel processing in the classroom.

R code to start a cluster

```
library(parallel)

detectCores()

cl = makeCluster(2)
do.call("rbind",clusterCall(cl,function(cl)
     Sys.info()["nodename"]))

stopCluster(cl)
```

Examples:

- Sum and Mean
- Matrix Multiplication
- Linear Regression, model fitting and verification
- Bootstrapping
- Clustering
- Bayesian MCMC

Sum:

The calculation of the sum of a list of numbers $x_1, x_2, ..., x_n$. If $n = k * m$, then the list of n numbers can be represented as k subsets of m numbers. The calculation of the total can be represented as follows.

$$\begin{aligned} S_n &= \sum_{i=1}^{n} x_i && (1) \\ &= \sum_{j=1}^{k} \sum_{i=1}^{m} x_{ji} && (2) \\ &= S_{1m} + S_{2m} + ... + S_{km} && (3) \end{aligned}$$

Mean:

Exercise: How can the sample mean be computed in parallel?

Answer: Very clear what the answer is. However, it is very instructive.

Question: What other parallel computations could be possible?

Pooled Standard Deviation, ANOVA

R code for Sum:

```
x = c(1,12,23,34,45,56)

x.mat = matrix(x,3,2); x.mat

apply(x.mat, 2, sum)

sum(apply(x.mat, 2, sum))
```

R code for the parallel computation of the Sum:

```
library(parallel)

detectCores()

cl = makeCluster(2)
do.call("rbind",clusterCall(cl,function(cl)
     Sys.info()["nodename"]))

# parallel apply

parApply(cl, x.mat, 2, sum)
sum(parApply(cl, x.mat, 2, sum))
```

R code for the parallel computation of Matrix Multiplication:

```
A <- matrix(rnorm(1000000), 1000)

system.time(A %*% A)

library(snow)

cl = makeCluster(2)
do.call("rbind",clusterCall(cl,function(cl)
    Sys.info()["nodename"]))

system.time(parMM(cl, A, A))

stopCluster(cl)
```

Linear Regression Simulation Study and Bootstrapping.

See Vinh Q. Nguyen's 2009 presentation at Department of Statistics, University of California, Irvine

Parallel Computing with R using snow and snowfall

`www.ics.uci.edu/~vqnguyen/talks/ParallelComputingSeminaR.pdf`

Very nice simulation study and Simple Linear Regression Bootstrap!

Checking the impact of violations of the assuming of normal errors.

Bootstrapping

```
library(boot)
```

```
boot(data, statistic, R, sim = "ordinary", stype = c("i", "f",
     "w"), strata = rep(1,n), L = NULL, m = 0, weights = NULL,
     ran.gen = function(d, p) d, mle = NULL, simple = FALSE, ...,
     parallel = c("no", "multicore", "snow"),
     ncpus = getOption("boot.ncpus", 1L), cl = NULL)
```

Bootstrapping

```
library(multicore)

system.time(sales.decline.boot <- boot(sales, decline.means,
     R = 60000, stype="f",strata=sales[,2],
     parallel = c("multicore"), ncpus = 2))
```

Bootstrapping

```
# Setup the cluster on 2 nodes with 6 cpus

library(Rmpi)
library(snow)

cl = makeCluster(6)
do.call("rbind",clusterCall(cl,function(cl)
     Sys.info()["nodename"]))

library(rlecuyer)
clusterSetupRNG(cl)
clusterEvalQ(cl,library(boot))

system.time(sales.decline.boot <- boot(sales, decline.means,
     R = 10000, stype="f",strata=sales[,2],
     parallel = c("snow"), ncpus = 6, cl = cl))
```

Clustering:

Try Revolution Analytics R.

RevoScaleR package

`rxKmeans`

`http://www.r-bloggers.com/k-means-clustering-on-big-data/`

Bayesian MCMC:

BUGSparallel project

SAS:

```
OPTIONS THREADS=YES CPUCOUNT=2;
```

Conclusions:

- Open Question: Is it a good idea to incorporate an introduction to parallel processing in an introductory Statistics class for Computer Scientists?
- Think of a good answer for your students.
- Not hard to introduce parallel processing in R.
- Might be the idea that attracts students to Statistics
- Lots of good examples on the Internet. See google.
- Give Revolution Analytics R a try. Very good for big data.

References:

McCallum, Q. Ethan and Weston, Stephen (2012) Parallel R, Data Analysis in a Distributed World. O'Reilly.