Big data analysis with bigmemory

Feng Li
feng.li@cufe.edu.cn

School of Statistics and Mathematics
Central University of Finance and Economics

May 13, 2014

Revision: May 13, 2014

Today we are going to learn...

The R's frustration on memory

- A numeric matrix containing 100 million rows and 5 columns consumes approximately 4 GB of memory in R.
- C/C++ or Fortran allow quick, memory-efficient operations on massive data sets, without the memory overhead of many R operations.
- Unfortunately, these languages are not well-suited for interactive data exploration, lacking the flexibility, power, and convenience of R's rich environment.

The `bigmemory` package

- The package `bigmemory` bridges the gap between R and C++, implementing massive matrices in memory and supporting their basic manipulation and exploration.
- It supports matrices of double, integer, short, and char data types.
- In Linux environments, the package supports the use of shared memory for matrices with transparent read and write locking (Good!).

We are far behind

- At Google, the statistians use 64-bit Linux workstations with up to 32 GB of RAM to study massive data.
- We are typical at computers have 1-2 GB of random access memory and some still run 32-bit operating systems (Windoz!!!!).

The default R data frame

- Data frames and matrices in R were designed for data sets much smaller in size than the computer's memory limit.
- They are flexible and easy to use, with typical manipulations executing quickly on smaller data sets.
- The bigmemory package addresses a category of data sets. These can be massive data sets but not larger than the total available RAM.
- In some cases, a traditional data frame or matrix might suffice to store the data, but there may not be enough RAM to handle the **overhead** of working with a data frame or matrix.

The bigmemory package

- Multiple processors on the same machine can share access to the same copy of the massive data set, and subsets of rows and columns may be extracted quickly and easily for standard analyses in R.
- Transparent read and write locks provide protection from well-known pitfalls of parallel programming.
- Most significantly, R users of bigmemory donâĂŹt need to be C++ experts (and donâĂŹt have to use C++ at all, in most cases).
- And C++ programmers can make use of R as a convenient interface, without needing to become experts in the environment.

Use bigmemory package

```
> library(bigmemory)
> x <- read.big.matrix("ALLtraining.txt", sep = "\t", type = "intege
+ shared = TRUE, col.names = c("movie", "customer", "rating",
+ "year", "month"))
> dim(x)
```

Use biglm with bigmemory

- Using `biglm` package with `bigmemory` is provided via the `biglm.big.matrix()` and `bigglm.big.matrix()` functions

```
> lm.0 = biglm.big.matrix(rating ~ year, data = x, fc = "year")
> print(summary(lm.0)$mat)
```

Shared memory via parallel package

```
> library(parallel)
> cl <- makeSOCKcluster(c("localhost", "localhost", "localhost"))
> parSapply(cl, 1:5, worker, xdescr)
> stopCluster(cl)
```

Further suggested read

The R Package bigmemory: Supporting Efficient Computation and Concurrent Programming with Large Data Sets.