A Divide-and-conquer Strategy to Solve the Out-of-memory Problem of Processing Thousands of Affymetrix Microarrays

ChiaJu Lee, Dong Fu, Simon Lin, Pan Du, Hongmei Jiang, and Warren Kibbe
Robert H. Lurie Comprehensive Cancer Center, Department of Statistics, and Computational Biology and Bioinformatics Program, Northwestern University

Abstract

Extremely large amount of microarray data has been produced and accumulated. As of October 2007, 173,486 arrays have been deposited into the NCBI GEO database. It is anticipated that more information can be discovered from large-scale experiments or large compilations of experiments than from small experiments with just a handful arrays. However, processing such a huge volume of data requires tremendous computing resources, which is far beyond the capacity of most research labs. A frequently encountered issue is the out-of-memory problem when processing thousands of CEL files generated by the Affymetrix platform using Bioconductor. We propose a divide-and-conquer strategy to solve this problem. It works recursively by breaking down a problem into many sub-problems of the same type, until they become simple enough to be solved directly in the memory. The solutions to the sub-problems are then combined to give a solution to the original problem. We used the CAMDA 2007 META-analysis data set, which contains 5,896 microarrays, to test our approach. The results were validated against a golden standard data set obtained by using a main frame computer, i.e., to run 5,896 arrays on a computer with 1TB of physical memory. In summary, this study is aimed to develop a general strategy to run any established pre-processing algorithms of Affymetrix in the Bioconductor package on a commodity computer cluster (32-bit CPU and 1GB of memory for each CPU).

Key words: divide-and-conquer, out-of-memory, Affymetrix arrays, R/Bioconductor

Introduction

R/Bioconductor is one of the most frequently used computational packages for microarray data analysis (Gentleman, et al., 2004). It includes a variety of open-source libraries to address the issues of background correction, normalization, expression index summarization, and quality assessment (Irizarry, et al., 2003). However, R, by design, is not efficient to handle large data sets. Moreover, most of the Bioconductor libraries were developed with a test data set of only dozens of microarrays. Thus, we are interested to find out the computational limit of R/Bioconductor using the 5,896 microarrays in the CAMDA 2007 META-analysis data set and to find a general solution to the out-of-memory problem without rewriting individual libraries in R/Bioconductor.

Out-of-memory is one of the most frequently encountered problems when using R/Bioconductor

One of the most frequently encountered problems of using R/Bioconductor to analyze a large data set is the following error message:

```
Error: cannot allocate vector of size 280,470 Kb.
```

Consequently, it has been recurrently asked and discussed at the R and Bioconductor mailing list. For instance, the problem has been discussed 175 times at the Bioconductor mailing list between January 2007 and September 2007.

Current attempts to solve this problem include:

- To increase the memory allocation that can be used by R and to increase the virtual memory. However, this solution is limited by the 2GB addressing capability of 32-bit CPUs.
- To increase memory usage efficiency by rewriting the library in C. For instance, the most popular method of RMA was re-implemented in C and called "justRMA". However, it is still limited by the maximum addressing capability of 32-bit CPUs. In addition, to rewrite all possible emerging methods is not a scalable solution.
- To redesign the memory-usage architecture of the affy library. This effort is represented by the new BufferedMatrix library to swap the memory with the disk storage. Individual libraries have to be written to take advantage of the BufferedMatrix architecture.
- Recently, there is a package called "aroma.affymetrix" in Bioconductor that tries to use the disk file system to process unlimited number of arrays by re-implementing RMA and MBEI. Based on limited documentation, it is unclear how the physical addressing limit of 32-bit CPU is dealt by this method.

We are seeking a different approach to find a general solution without rewrite individual libraries. We use the RMA function in the affy library to illustrate and test our approach.

A mainframe with 1TB of memory failed to run the affy library with 6,000 arrays

To illustrate the computational challenge, we have tested the CAMDA 2007 data set on a mainframe computer. The SGI Altix computer at National Center for Supercomputing Applications has 1 TB (= 1,000 GB) of physical memory. A 64-bit version of R was recompiled on this machine.

However, a memory problem was still encountered when loading 5,896 arrays. Debugging of the process suggested the limitation was caused by the design of the affy library in C. As such, we were only able to run about 4,000 arrays on this mainframe. The peak memory usage was about 72 GB.

This kind of mainframe machine is not universally accessible to most small labs. The most common computational resource is a 32-bit or 64-bit Linux cluster with 64 to 128 CPUs and up to 16GB memory. Although the theoretic addressing capability of 64-bit CPU is 16 exabyte, most of the commodity 64-bit computers have a hardware design limit of 16GB to 32GB. Thus, we seek a solution to run Bioconductor libraries under limited memory. Our solution was inspired by the miracle of bootstrap resampling used in computational statistics that relies on one's own resource to solve the problem.

The divide-and-conquer approach to large data sets

Divide-and-conquer is a classical solution to many large computational problems. In the case of microarrays, a simple divide-and-conquer will not generate an optimal answer, because the final result will be highly dependent on the initial order of the splitting. To solve this problem, we use a Monte Carlo method to resample the initial condition many times. Note that a calculation using the full combination of initial condition is usually infeasible. For instance, taking 40 arrays at a time out from 6,000 arrays will generate 1.4 * 10 ^ 103 possible combinations.

To test the feasibility of our approach, we designed a library in R/Bioconductor called divide-and-conquer RMA (dcRMA). The workflow of dcRMA is shown in figure 1 and described as follows.

An alternative implementation is discussed at the end of this abstract.

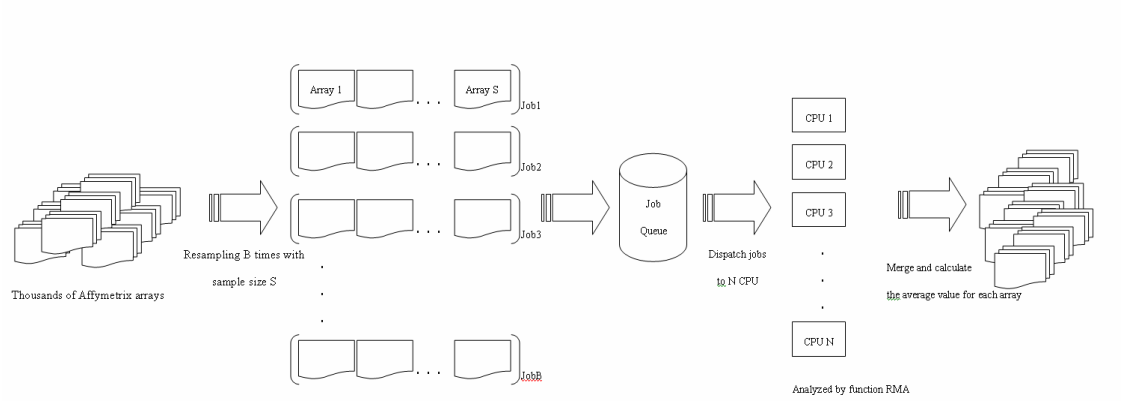


Figure 1. Flowchart of divide-and-conquer RMA (dcRMA).

Data set

The CAMDA 2007 META-analysis data set was used to test the divide-and-conquer approach. This dataset contains 5,896 arrays of diseased and normal human tissue samples and cell lines collected from ArrayExpress and GEO. All the samples were hybridized with the Affymetrix GeneChip Human Genome HG-U133A. For the proof-of-principle study, we selected a subset of 47 arrays. The study using all 5,896 arrays is on going and will be reported at the CAMDA conference.

dcRMA: Dividing steps

A resampling method is used to generate smaller subsets of arrays. Variable B is assigned as the number of resampling times, which represents the number of divided subsets of arrays. The number of resampling times should be reasonably large so that each one of the 5,896 arrays should be sampled at least once and included in at least one of the subset of arrays. In addition, we have a revolver algorithm to ensure each array is included at least once. Variable S is assigned as the number of arrays in a subset, i.e., the sample size of each subset of arrays. The choice of S is usually based on the maximum number of arrays that can be processed given the current memory limitation. We set B=100 and S=40 here for the test dataset of 47 arrays, i.e., the test data set are resampled to generate 100 subsets of arrays, each with sample size of 40.

dcRMA: Generating a job queue

Each subset of array data set is defined as a Job. All of these divided Jobs will be saved in the Job Queue and waiting for dispatching to an available CPU to do the RMA analysis. RMA is a procedure developed by Irizarry and colleagues to analyze Affymetrix data (Irizarry, et al., 2003). It includes background collection (mixture model), normalization (quantile), and summary (robust median fitting) all in one function. Following the tradition of MAS5, we rescale the median expression of each job to an arbitrarily chosen constant of *I*; it will implicitly address the issue of inter-job scale normalization. The allocation of each job to a different CPU was accomplished by the fork package that handles distributed processes. The variable N is assigned as the number of available CPUs; we set N=10 here. Once a Job is assigned to a CPU, the Job will be deleted from the Job Queue.

dcRMA: Summarization of expression measures of each array

Since the resampling times are reasonably large, each array is likely analyzed many times in different Jobs. Thus, the normalized value of S arrays in one Job will be saved separately. A matrix is then created for each array to merge and average the normalization values from different

jobs. This summarized value is taken as a final output, representing normalized expression measures of each array.

Validation of the Results

We used a mainframe computer to run the RMA procedure under the brute force of memory superiority. The result was used as a golden standard to compare the dcRMA results against.

First, we calculated the correlation coefficients between the dcRMA result and the golden standard for all genes on each array. The distribution of *r* was plotted in Figure 2. We observed a nearly identical (r=0.9999) result compared with the golden standard.

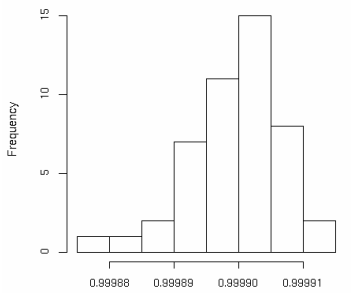
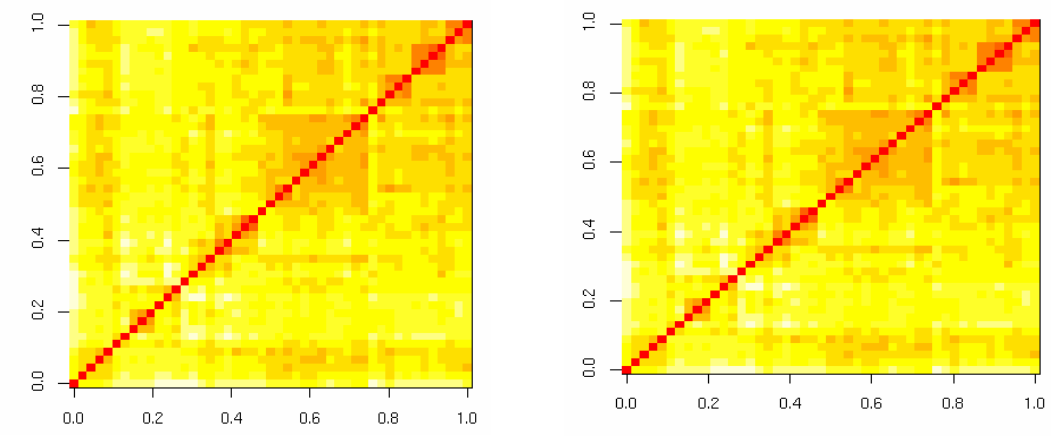


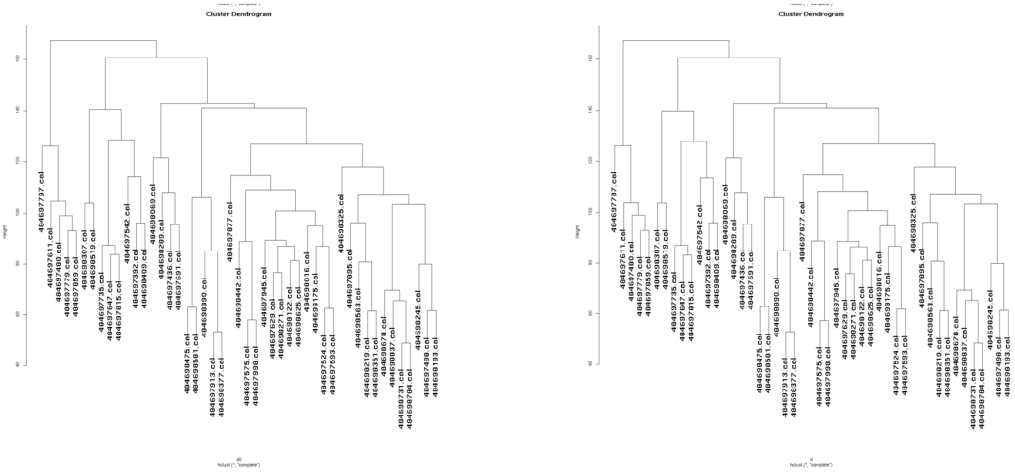
Figure 2. The distribution of correlation r between the dcRMA results and the golden standard. dcRMA was run under B=100 and S=40. The golden standard was calculated on a mainframe computer with 1TB of memory.

Because most of the machine learning methods, either unsupervised clustering or supervised models, rely on distance-based metrics, we further investigated if the small discrepancies between the dcRMA and the golden standard would make any difference. The inter-sample distances were calculated and visualized in Figure 3. Based on these distance matrices, a hierarchical clustering of the samples was conducted. We observed identical clustering results in Figure 4.



(A) (B)

Figure 3. Distance matrix between samples. (A) golden standard RMA (B) dcRMA



(A) (B)

Figure 4. Cluster of samples, based on data preprocessed by (A) golden standard RMA (B) dcRMA.

Conclusions and Ongoing Investigations

The advantage of our resampling approach over other method is its great simplicity - it is straightforward to apply the resampling method to any Affymetrix data summary libraries without rewriting the codes. The disadvantage of our approach is that while (under most conditions) it is asymptotically consistent, we are unable to theoretically approve it.

Same as bootstrapping methods, the number of resampling times, B, has to be investigated. We plan to run the procedure under B=4000, B=10000 and B=20000 and compare the results.

Moreover, to speed up the calculation and to reduce the potential number of B, an alternative implementation will be investigated:

1. Randomly shuffle the order of the microarrays.
2. Sequentially divide the microarrays into smaller jobs; each has S or fewer arrays.
3. Run each job using available CPU and memory resources.
4. Repeat step 1, 2 and 3. The results will be averaged. Stop if the number of iterations reaches B or some kind of convergence criteria is met.

References

Gentleman, R.C., Carey, V.J., Bates, D.M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A.J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J.Y. and Zhang, J. (2004) Bioconductor: open software development for computational biology and bioinformatics, *Genome Biol*, **5**, R80.

Irizarry, R.A., Bolstad, B.M., Collin, F., Cope, L.M., Hobbs, B. and Speed, T.P. (2003) Summaries of Affymetrix GeneChip probe level data, *Nucleic Acids Res*, **31**, e15.