

permGPU: Using graphics processing units in RNA microarray association studies

Ivo D. Shterev¹, Sin-Ho Jung^{1,2}, Stephen L. George^{1,2} and Kouros Owzar^{*1,2}

¹Department of Biostatistics and Bioinformatics, Duke University, 2424 Erwin Road, Durham, NC 27705, USA

²CALGB Statistical Center, Duke University, 2424 Erwin Road, Durham, NC 27705, USA

Email: Ivo D. Shterev - i.shterev@duke.edu; Sin-Ho Jung - sinho.jung@duke.edu; Stephen L. George - stephen.george@duke.edu; Kouros Owzar* - kouros.owzar@duke.edu;

*Corresponding author

Abstract

Background: Many analyses of microarray association studies involve permutation and bootstrap resampling, and cross-validation, that are ideally formulated as embarrassingly parallel computing problems. Given that these analyses are computationally intensive, scalable approaches that can take advantage of multi-core processor systems need to be developed.

Results: We have developed a CUDA based implementation, permGPU, that employs graphics processing units in microarray association studies. We illustrate the performance and applicability of permGPU within the context of permutation resampling for a number of test statistics. An extensive simulation study demonstrates a dramatic increase in performance when using permGPU on an NVIDIA GTX 280 card compared to an optimized C solution running on a conventional Linux server.

Conclusions: permGPU is available as an open-source stand-alone application and as an extension package for the R statistical environment. It provides a dramatic increase in performance for permutation resampling analysis in the context of microarray association studies. The current version offers six test statistics for carrying out permutation resampling analyses for binary, quantitative and censored time-to-event traits.

Background

Many resampling algorithms used in microarray association studies can be formulated within the framework of embarrassingly parallel problems in the sense that the algorithm can be split up into smaller components which can be completed mutually independently of each other. Standard algorithms used in this context include permutation and bootstrap resampling, and cross-validation. There are several protocols, including MPI and OpenMP, that facilitate parallelized programming for these types of algorithms.

Due to their highly parallel structure, Graphical Processing Units (GPU) are more effective than general-purpose CPUs for a set of algorithms widely used in the quantitative biomedical sciences. This has been demonstrated for example by using GPUs in feature detection in proteomics experiments [1], analysis of epistasis [2], statistical phylogenetics [3], and sequence alignment algorithms [4–7]. The R [8] extension package `gputools` [9] provides GPU enabled implementations of a set of commonly used functions for analysis of microarray data. Another attractive feature of using a GPU is that the hardware is relatively inexpensive, currently ranging from \$400 to \$1400, compared to high-end multi-core workstations or cluster farms. GPU hardware can be easily added to existing workstations.

In this paper, we present a Compute Unified Device Architecture (CUDA) framework, `permGPU`, that employs GPUs in microarrays association studies. We illustrate the performance and applicability of `permGPU` within the context of permutation resampling for a number of test statistics. The software is provided as a stand-alone application that can be used to carry out permutation resampling based in the case of binary (e.g., case versus control), quantitative (e.g., blood pressure) or censored time-to-event (e.g., time to death) traits. For wider use, we also have integrated `permGPU` into the R statistical environment.

Implementation

We illustrate our framework using a simulation study by implementing a single-step multiple testing procedure based on the maximum statistic as described in [10] and [11]. The CUDA toolkit from NVIDIA, a minimal set of extensions to the C and C++ languages, is used for programming on a GTX 280 GPU, with 240 processor cores and 1GB of memory. For comparison, we carry out a timing analysis based on a single CPU. The CPU code is compiled using `g++` version 4.3.2 with `-O3` and `-funroll-loops`, while the GPU code is compiled using `nvcc` with `-O2` and `--use_fast_math` optimization flags. Both the GPU and CPU analyses are carried out on a 2.83GHz Intel(R) Core(TM)2 Quad CPU Q9550 with 4GB RAM of memory using the AMD64 Linux operating system. For wider applicability to the research community, we

integrated `permGPU` into an R extension package. This implementation has been developed and tested on version 2.10.1.

The gene-expression matrix \mathbf{X} is of dimension $n \times K$, where K is the number of genes, or other features, and n is the number of patients. The vector of outcomes is denoted by \mathbf{Y} while the test statistic for testing the hypothesis of marginal association between feature k and the outcomes is denoted as T_k . We consider test statistics where the critical region is of the form $\{|T_k| > \xi\}$ for some $\xi > 0$. For a given family-wise error rate (FWER) $\alpha \in (0, 1)$, we determine the critical value $\xi > 0$ such that $P(\bigcup_{k=1}^K |T_k| > \xi) = \alpha$ under the hypothesis that no feature is associated with the outcome. The null sampling distribution is approximated using permutation resampling as follows:

1. Compute the K statistics T_1, \dots, T_K based on $\mathbf{Y}|\mathbf{X}^1, \dots, \mathbf{Y}|\mathbf{X}^K$.
2. Let $\tilde{\mathbf{Y}}$ be a random permutation of \mathbf{Y} .
3. Compute $\tilde{T}_{1,1}, \dots, \tilde{T}_{1,K}$, permutation replicates of the test statistics, based on $\tilde{\mathbf{Y}}|\mathbf{X}^1, \dots, \tilde{\mathbf{Y}}|\mathbf{X}^K$.
4. Compute $\zeta_1 = \max\{|\tilde{T}_{1,1}|, \dots, |\tilde{T}_{1,K}|\}$.
5. Repeat the last three steps $B - 1$ additional times.

The permutation FWER adjusted two-sided P -values are computed as $\tilde{p}_k^B = B^{-1} \sum_{b=1}^B I[|T_k| \leq |\tilde{T}_{b,k}|]$ and $\tilde{P}_k^B = B^{-1} \sum_{b=1}^B I[|T_k| \leq \zeta_k]$ respectively, where $I[\cdot]$ is the indicator function.

The code implementing our algorithm is a hybrid of kernels (GPU) and functions (CPU). The components of the code that compute the K test statistics, their maximum and P -values, are separate kernels. The kernel that computes the K test statistics is the most computationally expensive. Global memory reads and computation are the primary bottlenecks for speed. To increase global memory read speed, we allocate \mathbf{X} and other auxiliary data types via the function `cudaMallocPitch()`, thus assuring aligned memory access.

Findings

We illustrate the timing performance of our approach using an extensive simulation study considering the t test statistic, for two-sample problems, the Pearson test statistic, for continuous outcome, and a rank-covariance test statistic [12], for censored time to event outcome. The gene expression matrices are obtained by simulating $n \times K$ mutually independent and identically distributed standard normal variates where $n = 100, \dots, 1000$ and $K = 60000$. For the two-sample case, the groups are drawn from a Bernoulli law with mean 0.5. For the continuous case, the outcomes are drawn from a standard normal law. For the

time to event case, the expected censoring rate is set to 0.3. The illustrations for the t and Pearson test statistics are based on $B = 10,000$ permutations. The CPU approach for the rank-covariance statistic is prohibitively slow for large problems. For $(n, K) = (1000, 60000)$, an analysis based on a mere $B = 10$ replicates takes approximately 21 minutes versus only 16 seconds on the GPU. The CPU/GPU execution time ratios along with the GPU times (measured in seconds) are shown in Figure 1. It can be seen that the biggest speed increase is for the case of the survival test, where speedup factors of 78 can be observed.

Discussion

Although we have limited the discussions to three tests, our approach is more general. Currently, our code also implements the Wilcoxon, Spearman and Cox statistics, and can be extended using other test statistics including the family of score tests.

Permutation resampling to control FWER is one approach to address multiple testing for high-dimensional data. Our method can be easily extended to use the bootstrap via resampling with replacement. The false-discovery rate (FDR) [13] is another framework for adjusting for multiplicity. Our framework can be modified by omitting the calculation of the FWER adjusted P -values and applying any FDR algorithm to the unadjusted permutation P -values.

In many studies, primary interest is not the identification of significant features but rather the building of predictive models. It is neither appropriate nor practical to build the model using all features. Feature selection is typically used to identify, from the training data, a set of features, which are marginally important based on some criterion. Note that the feature selection needs to be redone for each cross-validation training set. Our framework can be customized to speedup the feature selection by recomputing the test statistic T_k based on the training set.

For microarray data sets the permutation analysis only needs to be done once and thus it might be argued that the gain in speed is not practically important. However, as illustrated in [14], for power and sample-size calculation, the permutation analysis needs to be repeated N times. Our approach can be extended to accommodate this type of analysis. For $(n, K) = (600, 60000)$ and $B = 10000$, our GPU Pearson algorithm takes about 12 seconds. A power analysis based on $B = 10000$ and $N = 1000$ would then be expected to take less than 4 hours. Since the projected speedup factor for this case is about 36, the expected time for completion on the CPU would exceed 5 days.

Conclusions

A CUDA based implementation for deploying GPUs in RNA microarray association studies has been presented. Our implementation can be customized by incorporating other statistical tests and scales readily with GPU cores. An extension for incorporating our framework into the R statistical environment has been developed. Dramatic increase in speed in comparison to an optimized C/C++ code was demonstrated. The increased speed becomes more pronounced when the sample size or number of markers is large, which makes our algorithm ideal for handling large genomic data sets. This is a practical framework that can be easily implemented using relatively inexpensive hardware.

Availability and requirements

- **Project Name:** permGPU
- **Project home page:** <http://code.google.com/p/permgpu/>
- **Operating System:** Linux AMD64.
- **Programming language:** Programming language: CUDA, C/C++ and R
- **Other requirements:** CUDA SDK and Toolkit 2.0 or higher; gcc/g++ 4.3.2; R (www.r-project.org) 2.10.1; Biobase (www.bioconductor.org) 2.6.1
- **License:** GPL v3.

Abbreviations

CPU: Central Processing Unit; CUDA: Compute Unified Device Architecture; GPL: General Public License; GPU: Graphics Processing Unit; FDR: False Discovery Rate; FWER: Family-Wise Error Rate; GTX 280: NVIDIA GeForce GTX 280; MPI: Message Passing Interface; OpenMP: Open Multi-Processing; RNA: Ribonucleic Acid; SDK: Software Development Kit

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

IDS conceptualized the research, designed, programmed, optimized and tested the algorithm, and drafted the manuscript; S-HJ contributed to the research and critically revised the manuscript; SLG contributed to the research, provided funding and critically revised the manuscript; KO proposed and conceptualized the research, and drafted the manuscript. All authors read and approved the final manuscript.

Acknowledgments

The authors thank John Pormann and Tom Milledge of the Duke Scalable Computing Support Center for providing GPU support.

References

1. Hussong R, Gregorius B, Tholey A, Hildebrandt A: **Highly accelerated feature detection in proteomics data sets using modern graphics processing units**. *Bioinformatics* 2009, **25**:1937–1943.
2. Sinnott-Armstrong NA, Greene CS, Cancare F, Moorel JH: **Accelerating epistasis analysis in human genetics with consumer graphics hardware**. *BMC Bioinformatics* 2009, **2**.
3. Suchard MA, Rambaut A: **Many-core algorithms for statistical phylogenetics**. *Bioinformatics* 2009, **25**:1370–1376.
4. Schatz MC, Trapnell C, Delcher AL, Varshney A: **High-throughput sequence alignment using Graphics Processing Units**. *BMC Bioinformatics* 2007, **8**.
5. Manavski S, G V: **CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment**. *BMC Bioinformatics* 2008, **9**.
6. Jung S: **Parallelized pairwise sequence alignment using CUDA on multiple GPUs**. *BMC Bioinformatics* 2009, **10**.
7. Liu Y, Maskell DL, B S: **CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units**. *BMC Bioinformatics* 2009, **2**.
8. R Development Core Team: **R: A Language and Environment for Statistical Computing**. R Foundation for Statistical Computing, Vienna, Austria 2009, [<http://www.R-project.org>]. [ISBN 3-900051-07-0].
9. Buckner J, Wilson J, Seligman M, Athey B, Watson S, Meng F: **The gputools package enables GPU computing in R**. *Bioinformatics* 2010, **26**:134–135.
10. Westfall PH, Young SS: *Resampling-Based Multiple Testing: Examples and Methods for P-value Adjustment*. New York: Wiley-Interscience 1993.
11. Ge Y, Dudoit S, Speed TP: **Resampling-based multiple testing for microarray data analysis**. *TEST* 2003, **12**:1–44.
12. Jung SH, Owzar K, George SL: **A multiple testing procedure to associate gene expression levels with survival**. *Statistics in Medicine* 2005, **24**:3077–3088.
13. Benjamini Y, Hochberg Y: **Controlling the false discovery rate: a practical and powerful approach to multiple testing**. *JR Statist Soc B* 1995, **57**:289–300.
14. Jung SH, Bang H, Young SS: **Sample size calculation for multiple testing in microarray data analysis**. *Biostatistics* 2005, **6**:157–169.

Figures

Figure 1 - CPU/GPU Time Ratios

Illustration of the CPU/GPU time ratio as a function of n , for the t , Pearson and rank-covariance [12] tests for $K = 60000$. For the t and Pearson tests $B = 10000$ permutations are used while for the rank-covariance test $B = 10$ permutations are used. The GPU times (in seconds) are also shown.

Additional Files

Additional file 1 — Supplementary Information for "permGPU: Using graphics processing units in RNA microarray association studies"

This document provides instructions for compiling and executing the code for the examples discussed in "permGPU: Using graphics processing units in RNA microarray association studies" by Shterev et al.

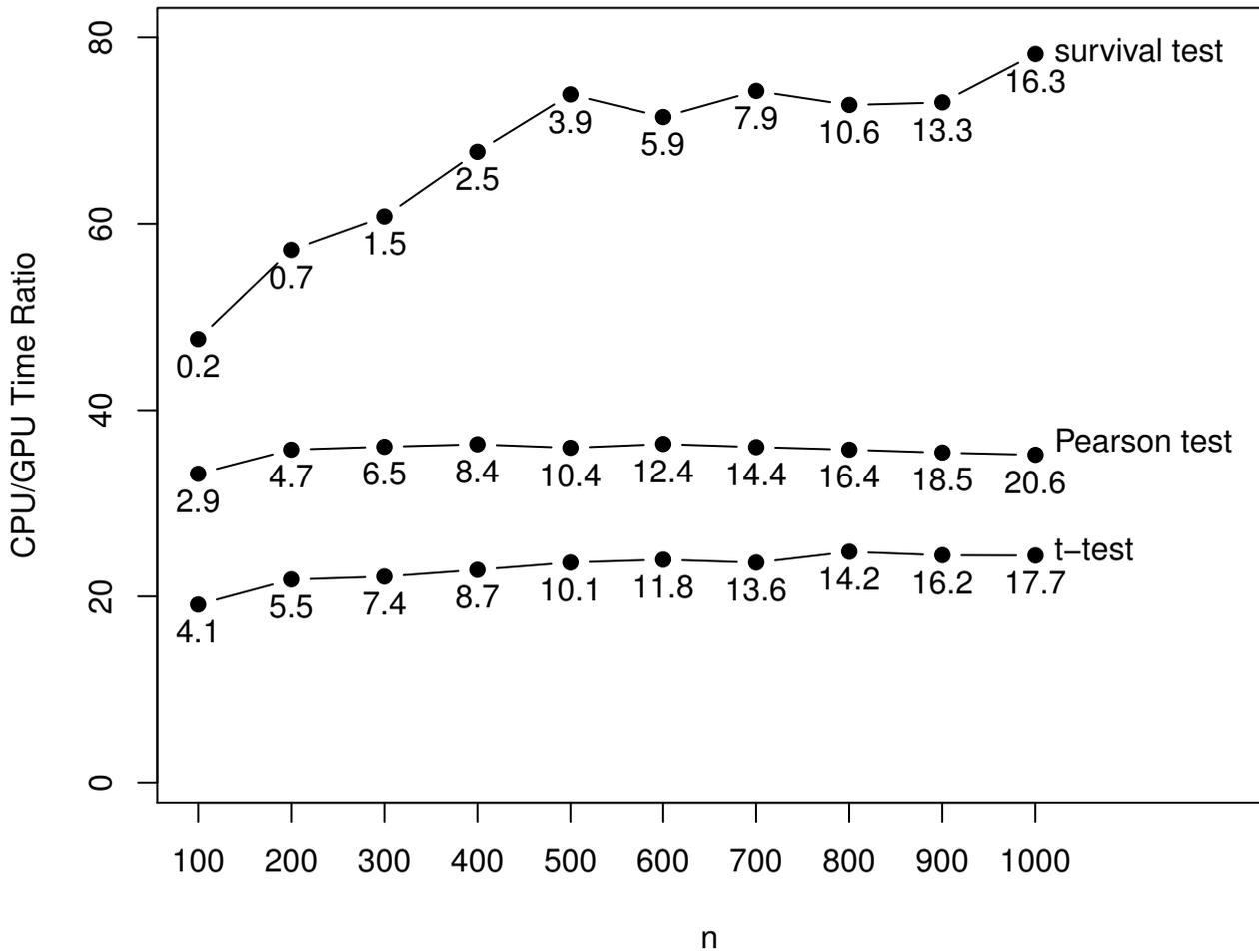


Figure 1

Additional files provided with this submission:

Additional file 1: permGPU_0.1.tar.gz, 27K

<http://www.biomedcentral.com/imedia/1724585569354884/supp1.gz>

Additional file 2: permGPU-standalone.zip, 40K

<http://www.biomedcentral.com/imedia/5327571303548841/supp2.zip>

Additional file 3: tutorial.pdf, 174K

<http://www.biomedcentral.com/imedia/1296338973548841/supp3.pdf>