BIOINFORMATICS APPLICATIONS NOTE Vol. 27 no. 16 2011, pages 2321–2322
doi:10.1093/bioinformatics/btr386

Data and text mining

Advance Access publication June 27, 2011

CAMPAIGN: an open-source library of GPU-accelerated data clustering algorithms

Kai J. Kohlhoff[1,*], Marc H. Sosnick[2], William T. Hsu[2], Vijay S. Pande[3,4,5] and Russ B. Altman[1,4,6]

[1]Department of Bioengineering, Stanford University, Stanford CA 94305-5448, [2]Department of Computer Science, San Francisco State University, 1600 Holloway Avenue, San Francisco, CA 94132-4163, [3]Department of Chemistry, [4]Department of Computer Science, [5]Department of Structural Biology and [6]Department of Genetics, Stanford University, Stanford CA 94305-5448, USA

Associate Editor: Martin Bishop

ABSTRACT

Motivation: Data clustering techniques are an essential component of a good data analysis toolbox. Many current bioinformatics applications are inherently compute-intense and work with very large datasets. Sequential algorithms are inadequate for providing the necessary performance. For this reason, we have created Clustering Algorithms for Massively Parallel Architectures, Including GPU Nodes (*CAMPAIGN*), a central resource for data clustering algorithms and tools that are implemented specifically for execution on massively parallel processing architectures.

Results: *CAMPAIGN* is a library of data clustering algorithms and tools, written in '*C for CUDA*' for Nvidia GPUs. The library provides up to two orders of magnitude speed-up over respective CPU-based clustering algorithms and is intended as an open-source resource.

New modules from the community will be accepted into the library and the layout of it is such that it can easily be extended to promising future platforms such as *OpenCL*.

Availability: Releases of the *CAMPAIGN* library are freely available for download under the LGPL from https://simtk.org/home/campaign. Source code can also be obtained through anonymous subversion access as described on https://simtk.org/scm/?group_id=453.

Contact: kjk33@cantab.net

Received on March 10, 2011; revised on May 30, 2011; accepted on June 8, 2011

1 INTRODUCTION

Data clustering algorithms have been useful in many fields of computer science and are of ever increasing importance in the life sciences. For instance, data analysis in modern day biology and bioinformatics often involves the extraction of patterns from diverse sources such as gene expression data and protein structures (Andreopoulos, 2009; Belacel, 2006; Chodera, 2008; Daxin, 2004; Zemla, 2007). This can require substantial amounts of processing time, putting a limit on the amount and quality of information that can be derived in reasonable time. To allow the development of more sophisticated analysis protocols, we present Clustering Algorithms for Massively Parallel Architectures, Including GPU Nodes (*CAMPAIGN*); a library of GPU-accelerated clustering algorithms for large-scale datasets. Equipped with an initial set of tools and GPU-ports of well-established algorithms, including K-means, K-centers and hierarchical clustering, *CAMPAIGN* is intended to form the basis for devising new parallel clustering codes specifically tailored to the GPU and other massively parallel architectures.

The '*C for CUDA*' parallel computing platform from Nvidia provides a framework for accessing CUDA-enabled graphics cards. The C-like syntax facilitates writing and maintaining code. We thus chose Nvidia's CUDA parallel computing engine for developing the first release of the library. Providing the code as individual modules allows easy modification and extensibility, for example by future modules using *OpenCL*.

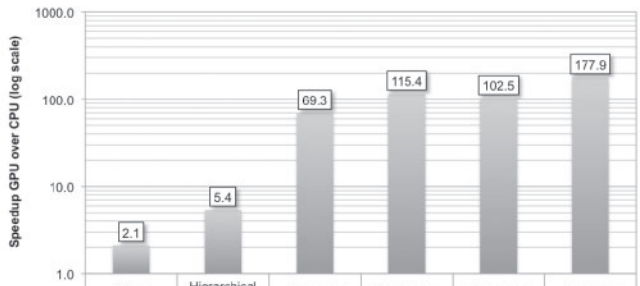
2 APPROACH

For the first release of the *CAMPAIGN* clustering library, we selected five popular clustering algorithms. For each algorithm, we implemented a serial CPU reference version and a GPU-accelerated version:

(1) K-means (and K_{ps}-means, a K-means variant for GPUs with parallel sorting for improved performance) (Lloyd, 1982).
(2) K-medoids (Ng, 1994).
(3) K-centers (a K-medoids variant in which medoids are placed only once according to a heuristic) (Dasgupta, 2005; Gonzalez, 1985).
(4) Hierarchical clustering (Hastie, 2009).
(5) Self-organizing map (Kohonen, 1982).

We initially included a selection of Euclidean, Manhattan and Chebyshev distance metrics. In general, we made an effort to ensure that the CPU and GPU versions produced the same result clusters. However, differences may occur because of rounding errors. For hierarchical clustering, substantial rearrangements of the dataset during run-time are required for our GPU version to be efficient. A CPU version that is intended to reproduce the same sequence of merging in the case of equidistant clusters is provided for comparison, but by default a more efficient sequential version is chosen. For K-means on the GPU, a basic implementation and an additional variant using parallel sorting for improved efficiency

*To whom correspondence should be addressed.



	SOM	Hierarchical clustering	K-means	Kps-means	K-medoids	K-centers
Speedup	2.1	5.4	69.3	115.4	102.5	177.9
Time on CPU	0:20:06	4:58:43	1:20:49	1:20:49	3:15:27	0:04:38
Time on GPU	0:09:31	0:54:50	0:01:10	0:00:42	0:01:54	0:00:02

Fig. 1. Performance comparisons of the current set of CAMPAIGN clustering algorithms including both K-means variants on a dataset of 9461 Affymetrix gene arrays of 20 099 human genes. The dataset's size in memory is 760 MB, with twice the requirement for algorithms using sorting (Kps-means and K-medoids). Total runtime (h:mm:ss) of the clustering algorithms for execution on CPU and GPU is given along with the factor of speedup achieved. The time for self-organizing-map is given for a single iteration; K-means converged after 39 iterations, and K-medoids was executed for 100 iterations. For hierarchical clustering, data matrix and access pattern for the CPU reference code were transposed, which improved performance of the CPU code. A strong dependence of the speedup achieved on the different clustering algorithms can be observed, which is indicative of each algorithm's suitability for parallel execution.

are available. Test data, examples and documentation as PDF and Doxygen-generated HTML files are provided.

3 RESULTS

We executed the parallel clustering codes on a single Nvidia Tesla C1060 GPU and compared the results with the serial versions running on one core of an Intel Xeon E5420 2.93 GHz CPU. We obtained performance improvements of one to two orders of magnitude for the *CAMPAIGN* GPU codes over sequential CPU reference implementations. Performance improvements over popular software packages such as MatLab were up to three orders of magnitude. In terms of accuracy, we find that for K-means, K-centers, K-medoids and hierarchical clustering, outputs for the different variants match closely with exception of machine-precision-related rounding errors, whereas for self-organizing map initial minor differences in rounding between two architectures are greatly amplified in successive steps.

We tested the algorithms with a biological dataset; a selection of 9461 gene array measurements of 20 099 human genes from NCBI GEO (Engreitz, 2010). A number of performance comparisons are summarized in Figure 1.

4 DISCUSSION

Although there is substantial interest in fast clustering algorithms, previously published implementations are in most cases not readily available as either binaries or source code. This lack of availability hinders the evolution and incremental improvement of algorithms. In addition, such black-box implementations can be frustrating to use or understand and cannot be customized. Besides offering ready-to-use implementations of GPU-based clustering algorithms, *CAMPAIGN* can serve both as a readily available benchmark against which to test future implementations of such algorithms, as well as a seed for the creation of a more exhaustive library of clustering codes. The current library is limiting the size of usable datasets to the amount of memory available on the GPU, such as 4GB in case of the Tesla GPU used here for testing, with less memory available on more low-end graphic cards.

5 CONCLUSION

We have introduced *CAMPAIGN*, an open-source library of data clustering algorithms and tools that aims to grow through contributions by the scientific and technical communities. The first batch of algorithms distributed with the initial release of the library offers one to two orders of magnitude speed-up as compared with CPU reference implementations.

ACKNOWLEDGEMENTS

The authors are grateful to Imran Haque and the staff at the Simbios NIH Center for Biomedical Computation for helpful suggestions.

Funding: National Institutes of Health through the NIH Roadmap for Medical research (U54 GM072970); a National Institutes of Health grant (LM-05652, to R.B.A.); Stanford School of Medicine Dean's Fellowship (to K.J.K.).

Conflict of Interest: none declared.

REFERENCES

Andreopoulos,B. *et al.* (2009) A roadmap of clustering algorithms: finding a match for a biomedical application. *Brief Bioinf.*, **10**, 297–314.

Belacel,N. *et al.* (2006) Clustering methods for microarray gene expression data. *OMICS,* **10,** 507–531.

Chodera,J.D. *et al.* (2007) Automatic discovery of metastable states for the construction of Markov models of macromolecular conformational dynamics. *J. Chem. Phys.,* **126**, 155101.

Dasgupta,S. and Long,P.M. (2005) Performance guarantees for hierarchical clustering. *J. Comput. Syst. Sci.*, **70**, 555–569.

Daxin,J. *et al.* (2004) Cluster analysis for gene expression data: a survey. *IEEE Trans. Knowl. Data Eng.*, **16**, 1370–1386.

Engreitz,J.M., *et al.* (2010) Independent component analysis: mining microarray data for fundamental human gene expression modules. *J. Biomed. Inform.,* **43**, 932–944

Gonzalez,T.F. (1985) Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, **38**, 293–306.

Hastie,T. *et al.* (2009) Hierarchical clustering. In *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, New York, 520–528.

Kohonen,T. (1982) Self-organized formation of topologically correct feature maps. *Biol. Cybern.*, **43**, 59–69.

Lloyd,S.P. (1982) Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, **IT-28**, 129–137.

Ng,R.T. and Han,J. (1994) Efficient and effective clustering methods for spatial data mining. In J.B.Bocca *et al.* (eds) *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago de Chile, Chile*. Morgan Kaufmann, pp. 144–155.

Zemla,A. *et al.* (2007) STRALCP–structure alignment-based clustering of proteins. *Nucleic Acids Res.*, **35**, e150.