MineBench: A Benchmark Suite for Data Mining Workloads

Ramanathan Narayanan[†] Berkin Özışıkyılmaz[†] Joseph Zambreno[§*] Gokhan Memik[†] Alok Choudhary[†]
[†]Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208, USA
{*ran310, boz283, memik, choudhar*}*@eecs.northwestern.edu*

[§]Electrical and Computer Engineering
Iowa State University
Ames, IA 50011, USA
zambreno@iastate.edu

Abstract

Data mining constitutes an important class of scientific and commercial applications. Recent advances in data extraction techniques have created vast data sets, which require increasingly complex data mining algorithms to sift through them to generate meaningful information. The disproportionately slower rate of growth of computer systems has led to a sizeable performance gap between data mining systems and algorithms. The first step in closing this gap is to analyze these algorithms and understand their bottlenecks. With this knowledge, current computer architectures can be optimized for data mining applications. In this paper, we present MineBench, a publicly available benchmark suite containing fifteen representative data mining applications belonging to various categories such as clustering, classification, and association rule mining. We believe that MineBench will be of use to those looking to characterize and accelerate data mining workloads.

1. Introduction

Data mining, a technique to understand and convert raw data into useful information, is increasingly being used in a variety of fields like marketing, business intelligence, scientific discoveries, biotechnology, internet searches, and multimedia. In data mining, the process of extracting information from raw data is automated, and mostly predictive. This is the aspect that makes data mining different from common database techniques. Data mining is essential to automate the finding of useful information from such large input data. For instance, a grocery chain would apply association rule mining to find and analyze local buying patterns.

Recent advances in data extraction techniques have resulted in massive data sets for data mining applications. A survey done by University of California, Berkeley indicates that an average person collects 800MB of data a year [22]. As a result, increasingly complex data mining algorithms are being used to extract information from these vast data sets. However, limitations in overall system performance will ultimately result in prohibitive execution times for these crucial applications. Hence there is a need to redesign modern architectures to optimize their performance on data mining workloads. This process involves two steps: First, since data mining is a relatively new application area, there is a need to understand the performance characteristics of various representative applications. Second, high performance computer architectures optimized for data mining systems need to be developed and their performance on commonly used workloads needs to be evaluated. For these purposes, there is a need for a benchmark that encompasses various representative data mining algorithms.

In this paper, we present MineBench, which contains fifteen representative data mining workloads from various categories. The workloads chosen represent the heterogeneity of algorithms and methods used in data mining. Applications from clustering, classification, association rule mining and optimization categories are included in MineBench. The codes are full fledged implementations of entire data mining applications, as opposed to stand-alone algorithmic kernels. We provide OpenMP parallelized codes for twelve of the fifteen applications. An important aspect of data mining applications are the data sets used. For most of the applications, we provide three categories of datasets with varying sizes: small, medium, and large. In addition, we provide source code, information for compiling the applications using various compilers, and command line arguments for all of the applications.

The remainder of this paper is organized as follows. In the following section, we provide a brief overview of the related work in this area. In Section 3, we discuss the data mining applications included in our benchmark suite. Section 4 details the various input data sets available. In Section 5, we provide command line arguments for the applications, discuss compiler compatibility, and simulation methods available. Finally the paper is concluded in Section 6 with a summary.

*This work was carried out when Joseph Zambreno was affiliated with the Department of EECS, Northwestern University

1-4244-0509-2/06/$20.00 ©2006 IEEE

Table 1. Overview of the MineBench data mining benchmark suite

Application	Category	Description
ScalParC	Classification	Decision tree classification
Naive Bayesian	Classification	Simple statistical classifier
K-means	Clustering	Mean-based data partitioning method
Fuzzy K-means	Clustering	Fuzzy logic-based data partitioning method
HOP	Clustering	Density-based grouping method
BIRCH	Clustering	Hierarchical Clustering method
Eclat	ARM	Vertical database, Lattice transversal techniques used
Apriori	ARM	Horizontal database, level-wise mining based on Apriori property
Utility	ARM	Utility-based association rule mining
SNP	Classification	Hill-climbing search method for DNA dependency extraction
GeneNet	Structure Learning	Gene relationship extraction using microarray-based method
SEMPHY	Structure Learning	Gene sequencing using phylogenetic tree-based method
Rsearch	Classification	RNA sequence search using stochastic Context-Free Grammars
SVM-RFE	Classification	Gene expression classifier using recursive feature elimination
PLSA	Optimization	DNA sequence alignment using Smith-Waterman optimization method

2. Related Work

Benchmarks play a major role in all domains. SPEC [19] benchmarks have been well accepted and used by several processor manufacturers and researchers to measure the effectiveness of their design. Other fields have popular benchmarking suites designed for the specific application domain: TPC [21] for database systems, SPLASH [23] for parallel architectures, and MediaBench [12] for media and communication processors.

Several bioinformatics benchmark suites have been developed recently. BioInfoMark [13], BioBench [3], and BioPerf [5] all contain several applications in common, including Blast, FASTA, Clustalw, and Hmmer. The bioinformatics applications presented in MineBench differ from those included in these benchmark suites. BioInfoMark and BioBench contain only serial workloads. In Bioperf, a few applications have been parallelized, unlike in MineBench which contains full fledged OpenMP parallelized codes of all bioinformatics workloads. Compared to the above benchmarks, MineBench covers a wider spectrum of data mining algorithms, in terms of quantity and diversity.

3. Benchmark Suite Overview

Data mining applications can be broadly classified into association rule mining, classification, clustering, data visualization, sequence mining, similarity search, optimization, and text mining, among others. Each domain contains unique algorithmic features. In establishing MineBench, we based our selection of categories on how commonly these applications are used in industry, and how likely they are to be used in the future. The fifteen applications that currently comprise MineBench are listed in Table 1, and are described in more detail in the following sections.

3.1 Classification Workloads

A classification problem has an input dataset called the training set, which consists of example records with a number of attributes. The objective of a classification algorithm is to use this training dataset to build a model such that the model can be used to assign unclassified records into one of the defined classes [10].

ScalParC is an efficient and scalable variation of decision tree classification [11]. The decision tree model is built by recursively splitting the training dataset based on an optimality criterion until all records belonging to each of the partitions bear the same class label. Decision tree based models are relatively inexpensive to construct, easy to interpret and easy to integrate with commercial database systems. ScalParC uses a parallel hashing paradigm to achieve scalability during the splitting phase. This approach makes it scalable in both runtime and memory requirements.

The *Naive Bayesian* classifier [8], a simple statistical classifier, uses an input training dataset to build a predictive model (containing classes of records) such that the model can be used to assign unclassified records into one of the defined classes. Naive Bayes classifiers are based on probability models that incorporate strong independence assumptions which often have no bearing in reality, hence the term 'naive'. They exhibit high accuracy and speed when applied to large databases.

Single nucleotide polymorphisms (SNPs), are DNA sequence variations that occur when a single nucleotide is altered in a genome sequence. Understanding the importance of the many recently identified SNPs in human genes has become a primary goal of human genetics. The *SNP* [7] benchmark uses the hill climbing search method, which selects an initial starting point (an initial Bayesian Network structure) and searches that point's nearest neighbors. The

neighbor that has the highest score is then made the new current point. This procedure iterates until it reaches a local maximum score.

Recent advances in DNA microarray technologies have made it possible to measure expression patterns of all the genes in an organism, thereby necessitating algorithms that are able to handle thousands of variables simultaneously. By representing each gene as a variable of a Bayesian Network (BN), the gene expression data analysis problem can be formulated as a BN structure learning problem. *GeneNet* [7] uses a similar hill climbing algorithm as in SNP, the main difference being that the input data is more complex, requiring much additional computation during the learning process. Moreover, unlike the SNP application, the number of variables runs into thousands, but only hundreds of training cases are available. GeneNet has been parallelized using a node level parallelization paradigm, where in each step, the nodes of the BN are distributed to different processors.

SEMPHY [7] is a structure learning algorithm that is based on phylogenetic trees. Phylogenetic trees represent the genetic relationship of a species, with closely related species placed in nearby branches. Phylogenetic tree inference is a high performance computing problem as biological data size increases exponentially. SEMPHY uses the structural expectation maximization algorithm, to efficiently search for maximum likelihood phylogenetic trees. The computation in SEMPHY scales quadratically with the input data size, necessitating parallelization. The computation intensive kernels in the algorithm are identified and parallelized using OpenMP.

Typically, RNA sequencing problems involve searching the gene database for homologous RNA sequences. *Rsearch* [7] uses a grammar-based approach to achieve this goal. Stochastic context-free grammars are used to build and represent a single RNA sequence, and a local alignment algorithm is used to search the database for homologous RNAs. Rsearch is parallelized using a dynamic load-balancing mechanism based on partitioning the variable length database sequence to fixed length chunks, with specific domain knowledge.

SVM-RFE [7], or Support Vector Machines - Recursive Feature Elimination, is a feature selection method. SVM-RFE is used extensively in disease finding (gene expression problem). The selection is obtained by a recursive feature elimination process - at each RFE step, a gene is discarded from the active variables of a SVM classification model, according to some support criteria. Vector multiplication is the computation intensive kernel of SVM-RFE, and data parallelism, using OpenMP, is utilized to parallelize the algorithm.

3.2. Clustering Workloads

Clustering is the process of discovering the groups of similar objects from a database to characterize the underlying data distribution [10]. It has wide applications in market or customer segmentation, pattern recognition, and spatial data analysis.

The first clustering application in MineBench is *K-means* [15]. K-means represents a cluster by the mean value of all objects contained in it. Given the user-provided parameter k, the initial k cluster centers are randomly selected from the database. Then, each object is assigned a nearest cluster center based on a similarity function. Once the new assignments are completed, new centers are found by finding the mean of all the objects in each cluster. This process is repeated until some convergence criteria is met. K-means tries to minimize the total intra-cluster variance.

The clusters provided by the K-means algorithm are sometimes called "hard" clusters, since any data object either is or is not a member of a particular cluster. The *Fuzzy K-means* algorithm [6] relaxes this condition by assuming that a data object can have a degree of membership in each cluster. Compared to the similarity function used in K-means, the calculation for fuzzy membership results in a higher computational cost. However, the flexibility of assigning objects to multiple clusters might be necessary to generate better clustering qualities. Both K-means and Fuzzy K-means are parallelized by distributing the input objects among the processors. At the end of each iteration, extra communication is necessary to synchronize the clustering process.

HOP [9], originally proposed in astrophysics, is a typical density-based clustering method. After assigning an estimation of the density for each particle, HOP associates each particle with its densest neighbor. The assignment process continues until the densest neighbor of a particle is itself. All particles reaching the same such particle are clustered into the same group. The advantage of HOP over other density based clustering methods is that it is spatially adaptive, coordinate-free and numerically straightforward. HOP is parallelized using a three dimensional KD tree data structure, which allows each thread to process only a subset of the particles, thereby reducing communication cost significantly.

BIRCH [26] is an incremental and hierarchical clustering algorithm for large databases. It employs a hierarchical tree to represent the closeness of data objects. BIRCH scans the database to build a clustering-feature (CF) tree to summarize the cluster representation. For a large database, BIRCH can achieve good performance and scalability. It is also effective for incremental clustering of incoming data objects, or when an input data stream has to be clustered.

In the next version of MineBench there will be more clustering applications incorporated from Clusbench [4]. This clustering benchmark consists of single threaded versions of K-means online, K-means batch, Self Organizing Map-1D (SOM-1D), SOM-2D, Hierarchical K-means online and Hierarchical SOM-1D algorithms. K-means batch algorithm is similar to the implementation available in MineBench. However, in K-means online algorithm, the cluster center

Table 2. Summary of MineBench Execution commands

Application	Execution Command
ScalParC	scalparc <datafile> <# records> <# attributes> <# classes> <# threads>
Naive Bayesian	bci [options] -d—-h <hdrfile> <tabfile> <bcfile>
K-means	example [options] -i <input file>
Fuzzy K-means	example [options] -f -i <input file>
HOP	para_hop <num particles> <particle file> <nsmooth> <bucket_size> <nHop> <nthreads>
BIRCH	birch <parafile> <schefile> <projfile> <datafile>
Eclat	eclat -i <input file> -o <output file> -s <support>
Apriori	no_output_apriori -i <input file> -f <offset file> -s <support> -n <# threads>
Utility	utility_mine <transaction file> <offset file> <profit file> <utility threshold> <# threads>
SNP	snp <input file>
GeneNet	genenet.icc <input file>
SEMPHY	semphy.mt -s <input file> -f <input format> -m <model type> -G <alpha value>
Rsearch	rsearch [options] <query sequence file> <data file>
SVM-RFE	svm_mkl <data file> <# cases> <# genes> <# iterations>
PLSA	parasw.mt <sequence 1> <sequence 2> <block height> <block width> <grid rows> <grid columns> <# alignments> <# threads>

positions are updated during each assignment of the objects. Hence it is an incremental version of batch K-means. The self-organizing map is a type of artificial neural network which uses unsupervised learning to reduce the dimensions of the data. The algorithm randomizes the input weight vectors for the input data. Then for each object, Euclidean distance is used to find similarity between the object and each node's weight vector in the map. Finally, the nodes in the neighborhood of the nearest map node are updated by pulling them closer to the input object. The difference between SOM-1D and SOM2-D is the way the original nodes weight vectors are connected in space. Hierarchical versions of this algorithm employ recursive calls of the original clustering algorithm. More information regarding these applications and their usage is presented by Altun et. al. [4].

3.3. ARM Workloads

The goal of Association Rule Mining (ARM) is to find interesting relationships hidden in large data sets. More specifically, it attempts to find the set of all subsets of items or attributes that frequently occur in database records [10]. In addition, the uncovered relationships can be represented in the form of association rules, which state how a given subset of items influence the presence of another subset.

Apriori [2] is the first ARM algorithm that pioneered the use of support-based pruning to systematically control the exponential growth of the search space. It is a level-wise algorithm that employs a generate-and-test strategy for finding frequent itemsets. It is based on the property that all non-empty subsets of a frequent itemset must all be frequent (the so-called "Apriori" property). For determining frequent items in a fast manner, the algorithm uses a hash tree to store candidate itemsets. Note: This hash tree has item sets at the leaves and hash tables at internal nodes. The Apriori algorithm is parallelized by distributing chunks of the input data among the processors. The master processor then gathers the local candidate itemsets, and generates globally frequent itemsets.

Utility mining [14] is another association rule-based mining technique where the assumption of uniformity among items is discarded. Higher "utility" itemsets are identified from a database by considering different values for individual items. The goal of utility mining is to restrict the size of the candidate set so as to simplify the total number of computations required to calculate the value of items. It uses the "transaction-weighted downward closure property" to prune the search space. The parallelization paradigm applied to Utility mining is the same as in Apriori, described above.

Eclat [24] uses a vertical database format. It can determine the support of any k-itemset by simply intersecting the id-list of the first two (k-1)-length subsets that share a common prefix. It breaks the search space into small, independent, and manageable chunks. Efficient lattice traversal techniques are used to identify all the true maximal frequent itemsets.

3.4. Optimization Workloads

Sequence alignment is an important problem in bioinformatics used to align DNA, RNA or protein primary sequences so as to emphasize their regions of similarity. It is used to identify the similar and diverged regions between two sequences, which may indicate functional and evolutionary relationships between them. *PLSA* [7] uses a dynamic programming approach to solve this sequence matching problem. It is based on the algorithm proposed by Smith and Waterman, which uses the local alignment to find the

Table 3. Runtimes of MineBench applications with different compilers (in seconds)

Application	ICC 7.0	ICC 8.1	ICC 9.1	GCC 2.96	GCC 3.2	GCC 3.4.5
ScalParC	55.49	56.30	56.54	68.07	52.21	52.60
Naive Bayesian	26.50	69.85	25.59	24.48	72.63	24.60
K-means	29.70	81.10	29.24	29.54	30.46	32.70
Fuzzy K-means	962.40	1375.42	625.86	949.68	938.21	937.88
HOP	26.45	29.53	27.11	33.16	31.85	31.00
Birch	C.E.†	C.E.	C.E.	15.86	C.E.	C.E.
Eclat	C.E.	92.63	30.50	33.96	113.59	33.86
Apriori	55.53	98.53	54.64	53/42	74.24	51.64
Utility	8.65	9.14	6.16	9.52	8.47	8.37
SNP	855.41	995.98	C.E.	C.E.	C.E.	C.E.
GeneNet	1166.29	698.23	C.E.	C.E.	C.E.	663.5*
SEMPHY	880.40	1422.31	1423.50	C.E.	1356.27	1234.52
Rsearch	742.81	714.65	835.28	1473.70	1546.06	1831.15
SVM-RFE	51.69	44.57	40.74	C.E.	45.65	40.37
PLSA	1648.6	2295.68	3035.40	3045.05	1716.62	1733.29

† *C.E.: Compilation Error* * Compiled using GCC3.3.4 instead of GCC3.4.5

longest common substring in sequences. PLSA uses special data structures to intelligently segment the whole sequence alignment problem into several independent subproblems, which dramatically reduce the computation necessary, thus providing more parallelism than previous sequence alignment algorithms.

4. Input Datasets

Input data is an integral part of data mining applications. The data used in our experiments are either real-world data obtained from various fields or widely-accepted synthetic data generated using existing tools that are used in scientific and statistical simulations. During evaluation, multiple data sizes were used to investigate the characteristics of the MineBench applications. For the non-bioinformatics applications, the input datasets were classified into three different sizes: small, medium, and large. For the ScalParC and Naive Bayesian applications, three synthetic datasets were generated by the IBM Quest data generator [1]. Apriori also uses three synthetic datasets from the IBM Quest data generator with a varying number of transactions, average transaction size, and average size of the maximal large itemsets. For HOP and Birch, three sets of real data were extracted from a cosmology application, ENZO [16], each having 61440 particles, 491520 particles and 3932160 particles.

A section of the real image database distributed by Corel Corporation is used for K-means and Fuzzy K-means. This database consists of 17695 scenery pictures. Each picture is represented by two features: color and edge. The color feature is a vector of 9 floating points while the edge feature is a vector of size 18. Since the clustering quality of K-means methods highly depends on the input parameter k, both K-means were executed with 10 different k values ranging from 4 to 13.

Utility mining uses both real as well as synthetic datasets. The synthetic data consists of two databases generated using the IBM Quest data generator. The first synthetic dataset is a dense database, where the average transaction size is 10; the other is a relatively sparse database, where average transaction size is 20. The average size of the potentially frequent itemsets is 6 in both sets of databases. In both sets of databases, the number of transactions varies from 1000K to 8000K and the number of items varies from 1K to 8K. The real dataset consists of only one database of size 73MB, where the average transaction length is 7.2.

Bioinformatics applications use datasets obtained from different biological databases. SNP uses the Human Genic Bi-Allelic Sequences (HGBASE) database containing 616,179 SNPs sequences. For GeneNet, Yeast microarray data was used as the input dataset. SEMPHY considers three datasets from Pfam database. The dataset for Rsearch, obtained from Washington University at St. Louis [18], uses a sequence of length 97 to search a database of size 100KB. SVM-RFE uses a benchmark microarray data set on ovarian cancer. This dataset contains 253 (tissue samples) x 15154 (genes) expression values, including 91 control and 162 ovarian cancer tissues with early stage cancer samples. For PLSA, nucleotides ranging from 30K to 900K length are chosen as test sequences [7].

5. Using MineBench

The applications in MineBench are written using C/C++ and the codes have been parallelized using OpenMP. Table 2 provides a summary of MineBench executables and command line arguments. The details about the data sets have been given in the previous section, and the detailed com-

mandline usage is provided in the source distribution.

In the current version, we make use of GCC v2.96, and Intel C/C++ (ICC) compiler version 7.0 to compile serial and parallel codes, respectively. We have also tried compiling the codes using different versions of ICC and GCC (for ICC v7.0, v8.1, v9.1 and GCC v2.95, v3.2, v3.4.5), on Redhat Linux, Redhat Linux Enterprise, Fedora Core and Suse distributions. Most of the applications compile properly, requiring only slight changes to header files and makefiles. The execution times using each compiler are presented in Table 3. To obtain the runtimes, we have executed the MineBench applications on a dual-processor Intel Xeon 2.8GHz machine using medium sized data sets. Amongst the GCC compilers, we see that the newest version performs the best for a vast majority of the applications. On the other hand, ICC v9.1 produces the fastest running times for around half the applications.

5.1 Simulation of MineBench Applications

Because of the way these applications have been parallelized using OpenMP, it is difficult to cross-compile and simulate on most widely-used computer architecture simulators. In addition, the long execution times of these applications prohibit detailed simulations. An architectural simulator that can be used to evaluate single-threaded workloads is PTLSim [4]. On this x86-64 cycle-accurate simulator, it is possible to run executables created with ICC, using the -openmp_stubs flag to create a single-threaded application binary. Another important feature of the simulator is its ability to switch between simulation and native execution modes using built-in function calls. By examining the applications' call graphs and inserting appropriate PTLSim function calls, we have been able to perform an in-depth study of the computationally intensive kernels. By doing so, we reduce the simulation time by several orders of magnitude. The binaries containing PTLSim function calls are also provided on our center's website [20].

6. Conclusion

Data mining applications are gaining significance as computationally intensive workloads, necessitating application-specific architectures to optimize their performance. A new data mining benchmark is required for workload characterization and architecture evaluation. In this paper, we presented MineBench, a diverse benchmark suite that covers a wide spectrum of data mining applications. More information about benchmark characteristics and workload characterization can be found in [17, 25]. We have provided optimized source codes, data sets of various sizes, and information for compiling and using the MineBench applications on various platforms. MineBench is freely available at our center's website [20].

Acknowledgments

This work was supported in part by National Science Foundation (NSF) under grants NGS CNS-0406341, IIS-0536994/002, CNS-0551639, CCF-0621443, CCF-0546278, and NSF/CARP ST-HEC program under grant CCF-0444405, and in part by the Department of Energy's (DOE) SCiDAC program (Scientific Data Management Center), number DE-FC02-01ER25485, DOE grants DE-FG02-05ER25683, and DE-FG02-05ER25691, and in part by Intel Corporation.

References

[1] R. Agrawal, A. Arning, T. Bollinger, M. Mehta, J. Shafer, and R. Srikant. The Quest data mining system. In *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining*, Aug. 1996.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo. Fast discovery of association rules. *Advances in Knowledge Discovery and Data Mining*, pages 307–328, 1996.

[3] K. Albayraktaroglu, A. Jaleel, X. Wu, M. Franklin, B. Jacob, C. Tseng, and D. Yeung. BioBench: A benchmark suite of bioinformatics applications. In *Proceedings of The 5th International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Mar. 2005.

[4] O. Altun, N. Dursunoglu, and M. F. Amasyali. Clustering application benchmark. In *To Appear in Proceedings of the International Symposium on Workload Characterization (IISWC)*, Oct. 2006.

[5] D. Bader, Y. Li, T. Li, and V. Sachdeva. BioPerf: A benchmark suite to evaluate high-performance computer architecture on bioinformatics applications. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, Oct. 2005.

[6] J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, 1981.

[7] Y. Chen, Q. Diao, C. Dulong, W. Hu, C. Lai, E. Li, W. Li, T. Wang, and Y. Zhang. Performance scalability of data-mining workloads in bioinformatics. *Intel Technology Journal*, 09(12):131–142, May 2005.

[8] P. Domingos and M. Pazzani. Beyond independence: Conditions for optimality of the simple bayesian classifier. In *Proceedings of the International Conference on Machine Learning*, 1996.

[9] D. Eisenstein and P. Hut. Hop: A new group finding algorithm for N-body simulations. *Journal of Astrophysics*, (498):137–142, 1998.

[10] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, Aug. 2000.

[11] M. Joshi, G. Karypis, and V. Kumar. ScalParC: A new scalable and efficient parallel classification algorithm for mining large datasets. In *Proceedings of the 11th International Parallel Processing Symposium (IPPS)*, 1998.

[12] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. MediaBench: A tool for evaluating and synthesizing multimedia and communications systems. In *Proceedings of 30th Annual International Symposium on Microarchitecture (MICRO)*, pages 330–335, Dec. 1997.

[13] Y. Li, T. Li, T. Kahveci, and J. Fortes. Workload characterization of bioinformatics applications. In *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 15–22, Sept. 2005.

[14] Y. Liu, W. Liao, and A. Choudhary. A two-phase algorithm for fast discovery of high utility itemsets. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, May 2005.

[15] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[16] M. Norman, J. Shalf, S. Levy, and G. Daues. Diving deep: Data management and visualization strategies for adaptive mesh refinement simulations. *Computing in Science and Engineering*, 1(4):36–47, 1999.

[17] B. Ozisikyilmaz, R. Narayanan, J. Zambreno, G. Memik, and A. Choudhary. An architectural characterization study of data mining and bioinformatics workloads. In *To Appear in Proceedings of International Symposium on Workload Characterization (IISWC)*, Oct. 2006.

[18] Sean Eddy's Lab. Rsearch software repository. http://www.genetics.wustl.edu/eddy/software, 2005.

[19] Standard Performance Evaluation Corporation. SPEC CPU2000 V1.2, CPU Benchmarks. Available at http://www.spec.org, 2001.

[20] The Center for Ultra-scale Computing and Information Security (CUCIS) at Northwestern University. NU-Minebench version 2.0. Available at http://cucis.ece.northwestern.edu, 2006.

[21] Transaction Processing Performance Council. TPC-H Benchmark Revision 2.0.0, 2004.

[22] University of California, Berkeley. The 'how much information?' project. Available at http://www2.sims.berkeley.edu/, 2000.

[23] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *Proceedings of the 22nd International Symposium on Computer Architecture (ISCA)*, pages 24–36, June 1995.

[24] M. Zaki. Parallel and distributed association mining: A survey. *IEEE Concurrency, Special Issue on Parallel Mechanisms for Data Mining*, 7(4):14–25, Dec. 1999.

[25] J. Zambreno, B. Ozisikyilmaz, J. Pisharath, G. Memik, and A. Choudhary. Performance characterization of data mining applications using MineBench. In *9th Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW)*, Feb. 2006.

[26] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *SIGMOD*, 1996.