NU-MineBench 2.0

CUCIS Technical Report CUCIS-2005-08-01
Center for Ultra-Scale Computing and Information Security
(CUCIS)
Northwestern University

NU-MineBench Team

Jayaprakash Pisharath
Ying Liu
Wei-keng Liao
Alok Choudhary
Gokhan Memik
Janaki Parhi

Document created by
Jayaprakash Pisharath

August 2005

1 Introduction

With the enhanced features in recent computer systems, increasingly larger amounts of data are being accumulated in various fields. Recent trends indicate that data being collected doubles every year. A survey done by Intel Corporation indicates that an average person collects 800MB of data a year [Cor05]. This does not include the scientific data collected by corporations and research institutions. Future systems are bound to be data intensive. Data would be processed either offline using data collection tools, or real-time using a streaming data model. Data analysis tools and frameworks are bound to become more sophisticated. Such complex analysis tools are bound to be performance-hungry due to the amount of data that they require to handle.

On the other hand, recent computing trends suggest that the system performance (data based on memory and I/O bound workloads like TPC-H) has been improving at a rate of 10-15% per year, whereas, the volume of data that is collected doubles every year. The important obstacle is the fact that the performance of computer systems is improving at a slower rate when compared to the increase in the data and the requirements of data analysis. Having observed this trend, researchers have focused on efficient implementations of different data mining algorithms. Among these, a major approach taken is the development of parallel and distributed versions of such algorithms. While these algorithms have been efficiently improved, the basic characteristics that define these algorithms remain under studied. Such information in turn can be utilized during the implementation of the algorithms and the design/setup of the computing systems. Understanding the performance bottlenecks is essential not only for processor designers to adapt their architectures to data mining applications, but also for programmers to adapt their algorithms to the revised requirements of applications and architectures. This forms the motivation for this work.

The rest of this report is organized as follows. Section 2 discusses the methodology used in this work to perform the characterization. The details of the workloads used in this characterization study (the workloads form the NU-MineBench benchmark suite) are provided in Section 3. The characterization results are not provided in this document to keep the discussion simple.

2 Methodology

The primary goal of this study is to gain an in-depth understanding of the characteristics of data mining applications. In other words, a detailed workload characterization of data mining applications is the primary goal of this work. This is achieved by studying data mining applications through benchmarking schemes. The ultimate goal is to assemble a benchmark that effectively represents data mining applications.

Benchmarks play a major role in all domains. SPEC [Sta01] 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 [Tra04] for database systems, SPLASH [WOT$^+$95] for parallel machine architectures, MediaBench [LPMS97] for media and communication processors. Benchmarks do not only play a role in measuring the relative performance of different systems. They also aid programmers in the specific domain in various ways. For example, a programmer implementing a new data mining application can compare the performance (in terms of output quality, scalability, and execution time) of the new application to the applications in the benchmarking suite. In addition, the programmer can use certain types of algorithms and programming styles from the applications in the existing suite.

Although there has been previous work analyzing individual data mining applications [BF98, KQH98], analyzing the behavior of a complete benchmarking suite will certainly give a better understanding of the underlying bottlenecks for data mining applications. This work analyzes data mining applications from many perspectives and presents the key characteristics of these applications. Another important aspect of this study is implementing and analyzing scalable versions of the benchmark applications. As the size of the available datasets and their high-dimensionality grow, high performance computers are becoming essential platforms to execute the data mining applications.

The first question that is addressed by this work is the uniqueness of data mining applications. If they are unique, the next task lies in identifying the characteristics that make them distinct from other existing applications and domains. In order to answer these questions, this work takes up an extensive characterization study. The various perspectives of the characterization study is shown in Figure 1. Like traditional studies, applications are studied from an algorithmic perspective by analyzing their execution times, run times (order of execution) and other high level metrics. Various system characteristics involving runtime overheads, system

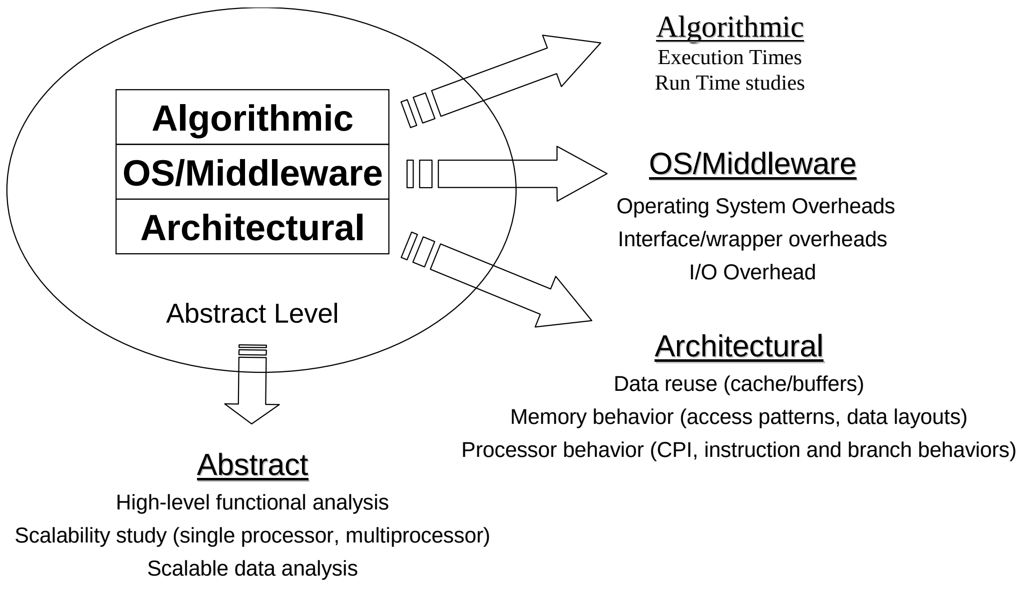
Algorithmic
OS/Middleware
Architectural
Abstract Level
Algorithmic
Execution Times
Run Time studies
OS/Middleware
Operating System Overheads
Interface/wrapper overheads
I/O Overhead
Architectural
Data reuse (cache/buffers)
Memory behavior (access patterns, data layouts)
Processor behavior (CPI, instruction and branch behaviors)
Abstract
High-level functional analysis
Scalability study (single processor, multiprocessor)
Scalable data analysis

Figure 1: The various perspectives of evaluation attempted in this characterization study

resources (locks), synchronization costs are also studied. A low-level architectural study is then performed extensively to understand the system architecture performance. This includes several components like the cache, processor, memory, and disks. Another important factor in data mining is the input data sizes. This study also characterizes the workload behavior based on input data sizes.

Scalability is deemed to be an important (and unavoidable) requirement in future systems. Scalability comes in many forms in both hardware and software. Traditionally, scalable versions of applications are developed by purely extending high-performance, parallel and distributed paradigms to the respective serial versions of the algorithms. This study analyzes various traditional scalability approaches as well. This is crucial since the final goal of this work is to enable development of high performnace data mining systems and algorithms. Hence, this work studies the scalability of applications based on processor resources and data sizes (dimensionality, etc) as well.

The following section describes the workload named as NU-MineBench that is developed and used for this study.

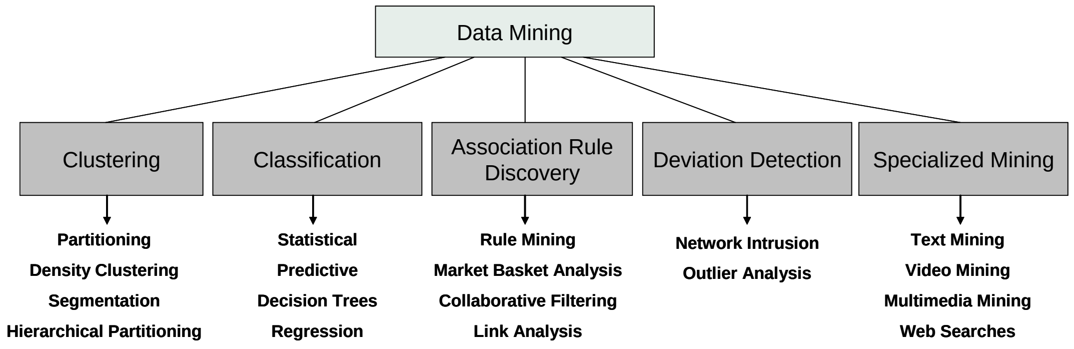

Figure 2: A data mining taxonomy. This leaf node of the graph shows well-known methodologies for performing data mining.

3 Data Mining Workload

Data mining, a technique to understand and present raw data, is becoming popular and is starting to be 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 OLAP and common database techniques. Data mining is essential to automatically find useful (predictive) information from such large input data. For instance, a cellular phone company would apply *clustering* to find a *probable* customer segment (regular travelers?) suitable for a new cellular phone plan.

Data mining applications are broadly classified into classification, clustering, association rule mining, sequence mining, similarity search, text mining, multimedia mining, and other categories based on the nature of their algorithms. Algorithmically, each domain is different. For instance, classification algorithms perform mining by building predictive models that represent data classes or concepts, whereas, clustering involves grouping a set of physical or abstract objects into categories of similar objects. Figure 2 shows a taxonomy of data mining applications (partly based on [HK00]). As can be seen, each data mining application differs by the nature of mining it performs.

A classification of data mining applications was shown in Figure 2. A subset of application domains of Figure 2 is used to establish NU-MineBench, a benchmarking suite containing well-known (and representative) data mining applications. The selection of categories as well as the applications in each category is based on how commonly these applications are used in industry and how likely to be used in the future, thereby achieving a realistic representation of the existing applications. Another concern of the algorithm selection is the scalability when executing on parallel or distributed systems.

NU-MineBench has applications from several domains and categories. The applications as well as important characteristics of the applications are listed in Table 1. Note that these are full-fledged application implementations of these algorithms (as against stand-alone algorithmic modules), which have been extensively optimized to remove all implementation inefficiencies. Also, the algorithms and the set of operations performed within them can be seen in commercial data mining tools, like Clementine (SPSS Inc.), Intelligent Data Miner (IBM Corporation) and SAS Enterprise Miner (SAS Institute Inc.).

There are several types of clustering algorithms in this study. The first clustering application in MineBench is K-means [Mac67]. K-means is a partition-based method and is arguably the most commonly used clustering technique. 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, K-means assigns each object to its nearest cluster center based on the similarity function. For example, for spatial clustering, usually the Euclid distance is used to measure the closeness of two objects. Once the assignments are completed, new centers are found by finding the mean of all the objects in each cluster. This process is repeated until two consecutive iterations generate the same cluster assignment. The clusters produced 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 [Bez81] relaxes this condition by assuming that a data object can have a degree of membership in each cluster. The Fuzzy K-means assigns each pair of object and cluster a probability. For each object, the sum of the probabilities to all clusters equals to 1. Compared to the Euclid distance used in K-means, the calculation for the fuzzy membership results in higher computational cost. However, the flexibility of assigning objects to multiple clusters might be necessary to generate better clustering qualities.

BIRCH [ZRL96] is one of the hierarchical clustering methods that employ a hierarchical tree to represent the closeness of data objects. BIRCH first scans the database to build a clustering-feature (CF) tree to summarize the cluster representation. Then, a selected clustering algorithm, such as K-means, is applied to the leaf nodes of the CF tree. For a large database, BIRCH can achieve good performance and scalability. It is also effective for incremental clustering of incoming

Table 1: Algorithms used in the study and their descriptions

Algorithms	***Category***	***Description***
k-Means	Clustering	Mean based data partitioning method
Fuzzy k-Means	Clustering	Fuzzy-logic based data partitioning method
BIRCH	Clustering	Hierarchical data segmentation method
HOP	Clustering	Density based grouping method
Naive Bayesian	Classification	Statistical classifier
ScalParC	Classification	Decision tree based classifier
Apriori	ARM	Horizontal database, level-wise mining based on Apriori property
Eclat	ARM	Vertical database, equivalence class based method
Utility	ARM	Utility based association rule mining method
SNP	Bayesian Network	Hill-climbing search method for DNA dependency extraction
GeneNet	Bayesian Network	Microarray based structure learning method for gene relationship extraction
SEMPHY	Expectation Maximization	Phylogenetic tree based structure learning method for gene sequencing
Rsearch	Pattern Recognition	Stochastic Context-Free Grammar based RNA sequence search method
SVM-RFE	Support Vector Machines	Recursive feature elimination based gene expression classifier
PLSA	Dynamic Programming	Smith Waterman optimization method for DNA sequence alignment

data objects.

Density-based methods grow clusters according to the density of neighboring objects or according to some other density function. HOP [EH98], originally proposed in astrophysics, is a typical density-based clustering method. After assigning an estimation of its 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 this state are clustered as a group. HOP is highly scalable when applied to large databases.

The Naive Bayesian classifier [DP96], 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. It is based on Bayes' Theorem. It is comparable in performance to decision tree based classification algorithms, and exhibits high accuracy and speed when applied to large databases. ScalParC, a scalable decision tree based classifier [JKK98], builds the decision tree by recursively splitting the training dataset based on an optimal criterion until all records belonging to each of the partitions bear the same class label.

Apriori [AMS$^+$96] is arguably the most influential association rule mining (ARM) algorithm. It explores the level-wise mining using the Apriori property: all nonempty subsets of a frequent itemset must also be frequent. Eclat [Zak99], another ARM algorithm, uses a vertical database format instead of the hash trees (horizontal format) as in apriori. This enables breaking the search space into small, independent, and manageable chunks. Efficient lattice traversal techniques are used to identify all the true maximal frequent itemsets.

Utility mining is another association rule based data mining technique where higher "utility" itemsets are identified from a database by considering different values of individual items as *utilities*. The work of utility mining to restrict the size of candidate set (so that memory and time usage can be reduced) and to simplify the total number of computations for calculation of utility or profit of items. The ultimate goal is to discover all the itemsets whose utility values are more than user specified threshold in a transaction database and to eliminate the itemsets having lower utility value [LLC05].

The primary goal of Bayesian network based methods is to build a learning network that represents the input data set. This is done by identifying the statistic relationship between the several variables present in the input data. A scoring function is introduced that evaluates a network with respect to the training data and outputs a value that reflects how well the network scores relative to the available data. Then the possible network structures are searched to find the best scored network, which usually is considered to be the network learned from the data. In general, the search problem is NP-hard, most algorithms use heuristic search methods, such as the MCMC (Markov Chain Monte Carlo) sampling, K2, Simulated Annealing etc., of which the greedy hill-climbing algorithm is the most efficient and popular approach [CDD$^+$05].

Single nucleotide polymorphisms (SNPs) are DNA sequence variations that occur when a single nucleotide is altered in the genome sequence. Identifying valid sequence variations is a goal of genomic research. This work focuses on a version of SNP that uses the hill climbing search method [CDD$^+$05]. This method first selects a specific point (an initial Bayesian Network structure) in the search space as the starting point. Algorithm then searches all the nearest neighbors for the current point in the search space, and then selects the neighbor that has the highest score as the new current point. This procedure iterates until no neighbor has higher score than the current point (i.e., reached a local maximum). GeneNet [CDD$^+$05] uses a similar hill climbing algorithm as in SNP. The difference here is that the input data is the microarray data, which requires lot more computations to perform the learning process. There are lot more variables used during the learning.

SEMPHY [CDD$^+$05] is a structure learning algorithm that is based on phylogenetic trees. Phylogenetic tree represents the genetic relationship of species by a tree where closely related species are placed in nearby branches. For DNA/protein sequences from different species, a phylogenetic relationship among them can be inferred to reflect the course of evolution. The goal of this algorithm includes searching for the best tree topology and the best branch lengths representing the distance between the two neighbors. Note that there are numerous branch length probabilities for each topology. SEMPHY uses Structural Expectation Maximization (probability estimation) algorithm to address this complication.

Typically, RNA sequencing problems involve searching the gene database for homologous RNA sequences. Rsearch [CDD$^+$05] uses a grammar based approach to achieve this goal. Rsearch uses SCFG (Stochastic Context-Free Grammar) to build and to represent a single RNA sequence with its secondary structure, and utilizes a local alignment algorithm named CYK algorithm, which is a decoding algorithm for SCFG, to search a database for homologous RNAs.

Support Vector Machines Recursive Feature Elimination (SVM-RFE) [CDD$^+$05] is a feature selection method that uses SVM techniques to refine and identify the optimum feature set in the feature data. It selects or omits dimensions of the data depending on a performance measure of SVM classifier. It is much more robust to data overfitting than other methods, including combinatorial search. SVM-RFE uses microarray data as the input data for analysis. SVM-RFE is used extensively in disease finding (gene expression). It eliminates gene redundancy automatically and yields better and more compact gene subsets. The selection is obtained by recursive feature elimination process: at each RFE step, a gene is discarded from the active variables of a SVM classification model. The features are eliminated according to a criterion related to their support to the discrimination function and the SVM is re-trained at each step.

Sequence alignment is an important tool in bioinformatics used to identify the similar and diverged regions between two sequences, e.g. biological DNA/protein sequences or text strings. PLSA [CDD$^+$05] uses a dynamic programming approach to solve this sequence (string) matching problem. It is based on the algorithm proposed by Smith and Waterman, which uses the local alignment to find the longest common substring in sequences. Since this method is dependent on the sequence length, it is computationally very intense.

3.1 Evaluation Datasets

Input data is an integral part of data mining applications. The data considered for the experiments are either real data got 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 NU-MineBench applications.

For non-bioinformatics applications, three input data were used in 3 different sizes: Small, Medium, and Large. For ScalParC and Nave Bayesian, three synthetic datasets (see Table 2 - “Classification”) were generated by the IBM Quest data generator [Sys04]. The notation Fx-Ay-DzK denotes a dataset with Function x, Attribute size y, and Data comprising of z*1000 records. Function 26 is a relatively complex function and produces large trees. Apriori and Eclat use three synthetic datasets from IBM Quest data generator (see Table 2 - “ARM”). D denotes the number of transactions, T is the average transaction size, and I is the average size of the maximal potentially large itemsets. In Table 2, the number of items is 1000 and the number of maximal potentially large itemsets is 2000. For HOP and BIRCH, three sets of real data were extracted from a cosmology application, ENZO [NSLD99], 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.

Table 2: Classification and Association Rule Mining Dataset Characteristics (Dataset size in MB).

Dataset	**Classification**		**ARM**	
	Parameter	(Size)	Parameter	(Size)
Small	F26-A32-D125K	(27)	T10-I4-D1000K	(47)
Medium	F26-A32-D250K	(54)	T20-I6-D2000K	(175)
Large	F26-A64-D250K	(108)	T20-I6-D4000K	(350)

Both K-means implementations use Euclid distance as the similarity function and execute it for the two features separately. Since the clustering quality of K-means methods highly depends on the input parameter k, both K-means were executed with ten different k values ranging from 4 to 13.

Utility mining uses both real datasets as well as synthetic datasets. The synthetic data constitutes of two databases generated using the IBM Quest data generator. The first synthetic dataset is a dense database, T10.I6.DX000K, where the average transaction size is 10; other is a sparse database, T20.I6.DX000K, 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 constitutes of only one database of size 73MB, where the average transaction length is 7.2.

For the bioinformatics applications, the datasets were provided by Intel Corporation [CDD$^+$05]. SNP uses the Human Genic Bi-Alletic Sequences (HGBASE) database [BLS$^+$00] containing 616,179 SNPs sequences. For GeneNet, the microarray data used for this study is assembled from [SSZ$^+$98]; they are the most popular cell cycle data of Yeast. SEMPHY considers three datasets from Pfam database [BCD$^+$04]. The three datasets (labelled in this work as S,M,L) and their characteristics is shown in Table 3. The software and the corresponding dataset for Rsearch were obtained from [Lab05]. The experiments use the sequence “mir-40.stk” with the length of 97 to search a part of database “Yeastdb.fa” with size of 100KB. SVM-RFE uses a benchmark microarray data set on ovarian cancer [AM02]. This dataset contains 253 (tissue samples) x 15154(genes) expression values, including 91 control and 162 ovarian cancer tissues with early stage cancer samples. For PLSA, nucleotides ranging from 30K to 900K length are chosen as test sequences. Since true sequences can seldom satisfy this specific size, some artificial sequences were used in the experiments [CDD$^+$05]. To make the ex-

Table 3: SEMPHY datasets and their characteristics

Dataset:	**S**	**M**	**L**
Taxa number	53	108	220
Sequence length	394	397	389

periments more comprehensive, several real DNA sequences were also chosen from a test suite provided by the bioinformatics group at Penn Stat University. The longest sequence pair used here is named TCR where the human sequence is 319,030 bp long and the mouse sequence is 305,636 bp long.

4 Summary

Data mining has become one of the most essential tools for various businesses as well as researchers in diverse fields. The surge in the operational speed of computing systems, and also the emergence of compact, low-cost, high-performance parallel and distributed systems have provided abundant venues for improving the performance of data mining algorithms. However, in recent years, there has also been a tremendous increase in the size of data that is collected and also the complexity of data mining algorithms themselves. The rate of this growth exceeds the rate of performance improvements in computing systems, thus widening the performance gap between data mining systems and algorithms. In this work, the goal is to narrow this gap by enabling designers to build systems that are tuned in accordance with the requirements and developments of data mining algorithms. This is achieved by performing a detailed characterization of a set of representative data mining programs from both the hardware and software perspectives. Several widely-used data mining algorithms from multiple categories were studied and then, a benchmark suite was designed. Named NU-MineBench, this benchmarking suite contains representative data mining applications. NU-MineBench suite includes applications from diverse applications ranging from cosmology, grocery stores to bioinformatics. The applications in NU-MineBench were evaluated using real systems and simulators.

References

[AM02] C. Ambroise and G. J. McLachlan. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proc. National Academy of Sciences*, 99(10):6562–6566, 2002.

[AMS$^+$96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. *Advances in knowledge discovery and data mining*, pages 307–328, 1996.

[BCD$^+$04] A. Bateman, L. Coin, R. Durbin, R.D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E.L.L. Sonnhammer, D.J. Studholme, C. Yeats, and S.R. Eddy. The pfam protein families database. *Nucleic Acids Research*, 32(Database):D138–D141, 2004.

[Bez81] J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, 1981.

[BF98] J. Bradford and J. Fortes. Performance and memory-access characterization of data mining applications. In *Workshop on Workload Characterization at the Annual International Symposium on Microarchitecture.*, 1998.

[BLS$^+$00] A.J. Brookes, H. Lehvaslaiho, M. Siegfried, J.G. Boehm, Y.P. Yuan, C.M. Sarkar, P. Bork, , and F. Ortigao. HGBASE: a database of snps and other variations in and around human genes. *Nucleic Acids Research*, 28(1):356–360, January 2000.

[CDD$^+$05] 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.

[Cor05] Intel Corporation. *Architecting the Era of Tera - Technical White Paper*, 2005. http://www.intel.com/technology/computing/archinnov/teraera/.

[DP96] P. Domingos and M. Pazzani. Beyond independence: Conditions for optimality of the simple bayesian classifier. In *Proc. of the Internaltional Conference on Machine Learning*, 1996.

[EH98] D.J. Eisenstein and P. Hut. Hop: A new group finding algorithm for N-body simulations. *Journal of Astrophysics*, (498):137–142, 1998.

[HK00] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, August 2000.

[JKK98] M.V. Joshi, G. Karypis, and V. Kumar. ScalParC: A new scalable and efficient parallel classification algorithm for mining large datasets. In *Proc. of the International Parallel Processing Symposium*, 1998.

[KQH98] J. Kim, X. Qin, and Y. Hsu. Memory characterization of a parallel data mining workload. In *Workshop on Workload Characterization: Methodology and Case Studies*, 1998.

[Lab05] Sean Eddy's Lab. Rsearch software repository, 2005. http://www.genetics.wustl.edu/eddy/software/.

[LLC05] Y. Liu, W.K. Liao, and A. Choudhary. A two-phase algorithm for fast discovery of high utility itemsets. In *Proc. of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2005.

[LPMS97] C. Lee, M. Potkonjak, and W.H. Mangione-Smith. Mediabench: A tool for evaluating and synthesizing multimedia and communicatons systems. In *Proc. of the International Symposium on Microarchitecture*, 1997.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[NSLD99] M.L. Norman, J. Shalf, S. Levy, and G. Daues. Diving deep: Data-management and visualization strategies for adaptive mesh refinement simulations. *Computing in Science and Engg.*, 1(4):36–47, 1999.

[SSZ$^+$98] P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, 1998.

[Sta01] Standard Performance Evaluation Corporation. *SPEC CPU2000 V1.2, CPU Benchmarks*, 2001.

[Sys04] IBM Almaden Research Center Intelligent Information Systems. Quest software, 2004. http://www.almaden.ibm.com/software/quest/Resources/index.shtml.

[Tra04] Transaction Processing Performance Council. *TPC-H Benchmark Revision 2.0.0*, 2004.

[WOT$^+$95] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *Proc. of the International Symposium on Computer Architecture*, 1995.

[Zak99] M.J. Zaki. Parallel and distributed association mining: A survey. *IEEE Concurrency, Special Issue on Parallel Mechanisms for Data Mining*, 7(4):14–25, December 1999.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *SIGMOD*, 1996.