An Architectural Characterization Study of Data Mining and Bioinformatics Workloads

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

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

***Abstract*— Data mining is the process of automatically finding implicit, previously unknown, and potentially useful information from large volumes of data. Recent advances in data extraction techniques have resulted in tremendous increase in the input data size of data mining applications. Data mining systems, on the other hand, have been unable to maintain the same rate of growth. Therefore, there is an increasing need to understand the bottlenecks associated with the execution of these applications in modern architectures. In this paper, we present MineBench, a publicly available benchmark suite containing fifteen representative data mining applications belonging to various categories: classification, clustering, association rule mining and optimization. First, we highlight the uniqueness of data mining applications. Subsequently, we evaluate the MineBench applications on an 8-way shared memory (SMP) machine and analyze important performance characteristics such as L1 and L2 cache miss rates, branch misprediction rates.**

I. Introduction

Data mining is a powerful technology that converts raw data into an understandable and actionable form, which can then be used to predict future trends or provide meaning to historical events. Originally limited to scientific research and medical diagnosis, these techniques are becoming central to a variety of fields including marketing and business intelligence, biotechnology, multimedia, and security. As a result, data mining algorithms have become increasingly complex, incorporating more functionality than in the past. Subsequently, there is a need for faster execution of these algorithms, which creates ample opportunities for algorithmic and architectural optimizations. In addition to the changing complexity of data mining algorithms, increasingly large amounts of data are being collected every year. Recent trends indicate that data collection rates are growing at an exponential pace. A survey done by Intel Corporation indicates that an average person collects 800MB of data a year [1]. Data mining is essential to extract useful information from such large amounts of data. However, limitations in overall system performance will ultimately result in prohibitive execution times for these crucial applications. Hence, there is a need to redesign and customize systems with respect to data mining applications. Considering the variety of data mining applications and their unique characteristics (cf. Section 1.1), this is a challenging task that cannot be accomplished through algorithmic analysis alone; the algorithmic analysis should be performed in combination with consideration of system bottlenecks. As data mining is a relatively new application area, very little is known in terms of the characteristics of the underlying computations and data manipulation, and their impact on computer systems.

The increasing performance gap between data mining systems and algorithms may be bridged by a two phased approach: a thorough understanding of the system characteristics and bottlenecks of data mining applications, followed by design of novel computer systems to cater to the primary demands of data mining workloads. We address this issue in this paper by investigating the execution of data mining applications on a shared-memory parallel (SMP) machine. We first establish a benchmarking suite of applications that we call MineBench, which encompasses many algorithms commonly found in data mining. We then analyze the architectural properties of these applications to investigate the performance bottlenecks associated with them.

A. Need for a New Benchmarking Suite

A new benchmarking suite is highly motivated if applications in a domain exhibit distinctive characteristics. In this section, we focus on the uniqueness of data mining applications, as compared to other application domains. We compare the architectural characteristics of applications across various benchmark suites. Specifically, data mining applications are compared against compute intensive applications, multimedia applications, streaming applications and database applications to identify the core differences. In this analysis, we used a variety of application suites including integer application benchmarks (SPEC INT from SPEC CPU2000 [2]), floating point application benchmarks (SPEC FP from SPEC CPU2000), multimedia application benchmarks (MediaBench [3]) and decision support application benchmarks (TPC-H from Transaction Processing Council [4]). We perform statistical analysis on 19 architectural characteristics (such as branch instructions retired, L1 and L2 cache accesses, etc.) of the applications and use this information to identify the core differences. Specifically, we monitor the performance counters of each application during execution using profiling tools, and obtain

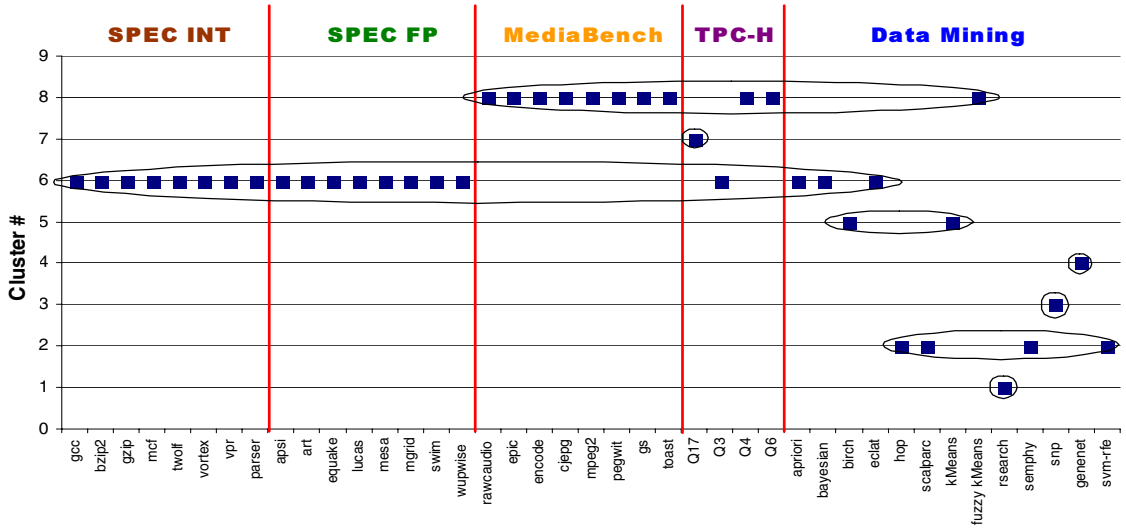

Fig. 1. Classification of data mining, SPEC INT, SPEC FP, MediaBench and TPC-H benchmark applications based on their characteristics. A K-means based clustering algorithm was used for this classification. Data mining applications tend to form unique clusters.

their individual characteristics. The experimental framework is identical to the one described in Section 4. The applications are then categorized using a K-means based approach, which clusters the applications with similar characteristics together. A similar approach has been used to identify a representative workload of SPEC benchmarks [5]. Figure 1 shows the scatter plot of the final configuration obtained from the results of the clustering method. Applications belonging to the SPEC INT, SPEC FP, TPC-H, and MediaBench benchmark suites are assigned to respective clusters. However, it can be seen that data mining applications stand out from other benchmark suites: they are scattered across several clusters. Although some of the data mining applications share characteristics with other application domains, they mostly exhibit unique characteristics. Another important property of the clustering results is the large variation within data mining applications. Although most of the applications in other benchmarking suites fall into one cluster, data mining applications fall into seven different clusters. This shows the large variation of characteristics observed in data mining applications. Overall, this analysis highlights the need for a data mining benchmark consisting of various representative algorithms that cover the spectrum of data mining application domains.

Table I shows the distinct architectural characteristics of data mining applications as compared to other applications. One key attribute that signifies the uniqueness of data mining applications is the number of data references per instruction retired. For data mining applications, this rate is 1.103, whereas for other applications, it is significantly less. The number of bus accesses originating from the processor to the memory (per instruction retired) verify the frequency of data access of data mining applications. These results solidify the intuition that data mining is data-intensive by nature.

The L2 miss rates are considerably high for data mining applications. The reason for this is the inherent streaming nature of data retrieval, which does not provide enough opportunities for data reuse. This indicates that current memory hierarchy is insufficient for data mining applications. It should be noted that the number of branch instructions (and the branch mispredictions) are typically low for data mining applications, which highlights yet another unique behavior of data mining applications.

Another important difference is the fraction of total instruction decodes to the instructions retired. This measure identifies the instruction efficiency of a processor. In our case, the results indicate that data mining applications are efficiently handled by the processor. The reason for this value being less than one is the use of complex SSE2 instructions. Resource related stalls comprises of the delay that incurs from the contention of various processor resources, which include register renaming buffer entries, memory buffer entries, and also the penalty that occurs during a branch misprediction recovery. The number of ALU operations per instruction retired is also surprisingly high for data mining applications, which indicates the extensive amount of computations performed in data mining applications. Therefore, data mining applications are computation-intensive in addition to being data-intensive.

What makes the data mining applications unique is this combination of high data rates combined with high computation power requirements. Such a behavior is clearly not seen in other benchmark suites. In addition, data mining applications tend to oscillate between data and compute phases, making the current processors and architectural optimizations mostly inadequate.

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 that are included in our benchmarking suite,

TABLE I

COMPARISON OF DATA MINING APPLICATION WITH OTHER BENCHMARK APPLICATIONS

Parameter†	*Benchmark of Applications*				
	SPECINT	**SPECFP**	**MediaBench**	**TPC-H**	**NU-MineBench**
Data References	0.81	0.55	0.56	0.48	1.10
Bus Accesses	0.030	0.034	0.002	0.010	0.037
Instruction Decodes	1.17	1.02	1.28	1.08	0.78
Resource Related Stalls	0.66	1.04	0.14	0.69	0.43
CPI	1.43	1.66	1.16	1.36	1.54
ALU Operations	0.25	0.29	0.27	0.30	0.31
L1 Misses	0.023	0.008	0.010	0.029	0.016
L2 Misses	0.003	0.003	0.0004	0.002	0.006
Branches	0.13	0.03	0.16	0.11	0.14
Branch Mispredictions	0.009	0.0008	0.016	0.0006	0.006

† *The numbers shown here for the parameters are values per instruction*

followed by the description of our methodology and input data sets in Section 4. In Section 5, we provide performance characterization results for single and multiprocessor cases. Finally, the paper is concluded in Section 6.

II. RELATED WORK

Benchmarks play a major role in all domains. SPEC [2] benchmarks have been well accepted and used by several chipmakers and researchers to measure the effectiveness of their designs. Other fields have popular benchmarking suites designed for the specific application domain: TPC [4] for database systems, SPLASH [6] for parallel architectures, and MediaBench [3] for media and communication processors.

Performance characterization studies similar to ours have been previously performed for database workloads [7], [8], with some of these efforts specifically targeting SMP machines [9], [10]. Performance characterization of individual data mining algorithms have been done [11], [12], where the authors focus on the memory and cache behavior of a decision tree induction program.

Characterization and optimization of data-mining workloads is a relatively new field. Our work builds on prior effort in analyzing the performance scalability of bioinformatics workloads performed by researchers at Intel Corporation [13]. As will be described in the following sections, we incorporate their bioinformatics workloads into our MineBench suite, and where applicable, make direct comparisons between their results and our own. However, MineBench is more generic and covers a wider spectrum than the bioinformatics applications previously studied [13]. Jaleel et al. examine the last-level cache performance of these bioinfomatics applications [14].

The bioinformatics applications presented in MineBench differ from other recently-developed bioinformatics benchmark suites. BioInfoMark [15], BioBench [16], BioSplash [17], and BioPerf [18] all contain several applications in common, including *Blast*, *FASTA*, *Clustalw*, and *Hmmer*. Srinivasan et al. [19] explore the effects of cache misses and algorithmic optimizations on performance for one of the applications in MineBench (SVM-RFE), while our work investigates several architectural features of data mining applications. Sanchez et al. [20] perform architectural analysis of a commonly used biological sequence alignment algorithm, whereas we attempt to characterize a variety of data mining algorithms used in biological applications.

III. BENCHMARK SUITE OVERVIEW

Data mining applications can be broadly classified into association rule mining, classification, clustering, data visualization, optimization, sequence mining, similarity search, and text mining, among others. Each domain contains unique algorithmic features. In establishing MineBench, we have selected applications from clustering, classification, association rule mining, and optimization. The selection of these categories is based on how commonly these applications are used in industry and academia, and how likely they are to be used in the future. The fifteen applications that currently comprise MineBench are listed in Figure 2, and are described in more detail in the following sections. Note that these are full-fledged application implementations of these algorithms (as opposed to stand-alone algorithmic modules), which have been extensively optimized to remove all implementation inefficiencies. It is necessary to study these applications in their entirety, as they are quite complex. A study that evaluates only kernels will not be able to identify several interesting features of these applications.

A. 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 [21].

ScalParC is an efficient and scalable variation of decision tree classification [22]. 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. Among many classification methods proposed over the years, decision trees are particularly suited

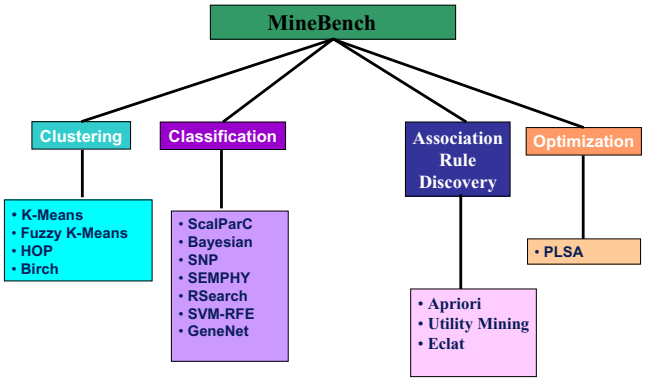

Fig. 2. Data mining applications in MineBench

for high-performance data mining, since they can be built relatively fast when compared to other methods.

The *Naive Bayesian* classifier [23], 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.

Single nucleotide polymorphisms (SNPs), are DNA sequence variations that occur when a single nucleotide is altered in a genome sequence. The *SNP* [13] benchmark uses the hill climbing local search method, which selects an initial starting point (an initial Bayesian Network structure) and searches all the neighbors of the starting point in the search space. The neighbor that has the highest score is then made the new current point. This procedure iterates until it reaches a local maximum score. *GeneNet* [13] 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.

SEMPHY [13] 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. This application uses a probability estimation algorithm to find the best tree topology and best branch lengths representing the distance between two neighbors.

Typically, RNA sequencing problems involve searching the gene database for homologous RNA sequences. *Rsearch* [13] 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.

SVM-RFE [13], or Support Vector Machines - Recursive Feature Elimination, is a feature selection method. SVM-RFE is used extensively in disease finding (gene expression). 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.

B. Clustering Workloads

Clustering is the process of discovering the groups of similar objects from a database to characterize the underlying data distribution [21]. It has wide applications in market or customer segmentation, pattern recognition, and spatial data analysis. The first clustering application in MineBench is *K-means* [24]. K-means represents a cluster by the mean value of all objects contained in it. Given the user-provided parameter k, the algorithm assigns the entities in the input data set among k clusters, according to a specific distance function. 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 [25] 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.

HOP [26], 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. *BIRCH* [27] is one of the hierarchical clustering methods that employ 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.

C. ARM Workloads

The goal of Association Rule Mining (ARM) is to find the set of all subsets of items or attributes that frequently occur in database records [21]. In addition, ARM applications extract rules regarding how a given subset of items influence the presence of another subset.

Apriori [28] is arguably the most influential ARM algorithm. It explores the level-wise mining of the property that all non-empty subsets of a frequent itemset must all be frequent (the so-called "Apriori" property). *Utility mining* [29] is another association rule-based mining technique where higher "utility" itemsets are identified from a database by considering different values of 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.

Eclat [30] 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

TABLE II
MINEBENCH EXECUTABLE PROFILES

Application	Instruction Count (billions)				Size (kB)
	1 Processor	2 Processors	4 Processors	8 Processors	
ScalParC	23.664	24.817	25.550	27.283	154
Naive Bayesian	23.981	N/A	N/A	N/A	207
K-means	53.776	54.269	59.243	77.026	154
Fuzzy K-means	447.039	450.930	477.659	564.280	154
HOP	30.297	26.920	26.007	26.902	211
BIRCH	15.180	N/A	N/A	N/A	609
Apriori	42.328	42.608	43.720	47.182	847
Eclat	15.643	N/A	N/A	N/A	2169
Utility	13.460	19.902	20.757	22.473	853
SNP	429.703	299.960	267.596	241.680	14016
GeneNet	2,244.470	2,263.410	2,307.663	2,415.428	13636
SEMPHY	2,344.533	2,396.901	1,966.273	2,049.658	7991
Rsearch	1,825.317	1,811.043	1,789.055	1,772.200	676
SVM-RFE	51.370	55.249	63.053	82.385	1336
PLSA	4,460.823	4,526.160	4,080.610	4,001.675	836

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.

D. Optimization Workloads

Sequence alignment is an important tool in bioinformatics used to identify the similar and diverged regions between two sequences. *PLSA* [13] 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 longest common substring in sequences.

IV. METHODOLOGY

In this section, we consider the applications in our MineBench suite, and distinguish the characteristics that make each application unique from both the algorithmic and the system perspective. We chose an Intel IA-32 multiprocessor platform for evaluation purposes. Our setup consists of an Intel PIII Xeon 8-way Shared Memory Parallel (SMP) machine running Red Hat Advanced Server 2.1. The system has 4 GB of shared memory. Each processor has a 16 KB non-blocking, integrated L1 data and instruction cache and a 1024 KB L2 cache.

For our experiments, we use the VTune Performance Analyzer [31] for profiling the functions within our applications, and for measuring the execution times. Using the VTune counters, we monitor a wide assortment of performance metrics: execution time, communication and synchronization complexity, memory behavior, and Instructions per Cycle (IPC) statistics. Each application was compiled with version 7.1 of the Intel C++ compiler for Linux. The applications have been parallelized using OpenMP, except Naive Bayesian, Birch and Eclat which are single threaded workloads.

A. 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 simulations. During evaluation, multiple data sizes were used to investigate the change of 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 benchmarks, three synthetic datasets were generated by the IBM Quest data generator [32]. Apriori and Eclat also use 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 [33], each having 61440 particles, 491520 particles and 3932160 particles, respectively.

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 point values while the edge feature is a vector of size 18. Both K-means implementations use Euclidean 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 and Fuzzy 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.

For the bioinformatics applications, the datasets were provided by Intel [13]. SNP uses the Human Genic Bi-Alletic Sequences (HGBASE) database [34] containing 616,179 SNPs sequences. For GeneNet, the microarray data used for this study is assembled from [35]; they are the most popular cell cycle data of Yeast. SEMPHY considers three datasets from the Pfam database [36]. The software and the corresponding dataset for Rsearch were obtained from [37]. 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 dataset on ovarian cancer [38]. 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 in length from 30K to 900K are chosen as test sequences. Since true sequences can seldom satisfy this specific size, some artificial sequences were used in the experiments [13]. To make the experiments more comprehensive, several real DNA sequences were also chosen from a test suite provided by the bioinformatics group at Penn State 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.

V. Architectural Characterization

The work in this section builds upon our previous workload characterization of the MineBench applications [39] for 8 processors. Due to time and space constraints, only medium sized datasets results have been presented in the following sections.

A. Execution Time and Scalability

In Table II, we present the total number of instructions executed across all processors along with the size of the executables. We can see that these benchmarks execute from tens of billions to thousands of billions of instructions. As the number of processors increases, the number of instructions executed is expected to increase due to the overhead of parallelization (locks, communication, synchronization etc.). However we observe that in some of the applications, instructions retired decreases as the number of processors increases. This may happen when the convergence criteria is reached at an earlier stage during execution of the parallel application. In our study, the usage of Vtune Performance Analyzer enables us to examine the characteristics of program execution across all execution phases, something that would not be feasible using simulation for applications of this size.

Figure 3 shows the benchmark application execution speedups when running on multiple processors. The performance numbers for the 2-processor case shows some trivial performance improvement for clustering and ARM workloads, while most of the remaining workloads perform slightly better or worse than the serial case. On the other hand, several benchmarks show good scalability with higher number of processors. When running on 8 processors, ScalParC executes 4.84 and 5.64 times faster than the 1 processor case for the small and large data sets, respectively. The best speedup, 7.55x on 8 processors, is seen in Utility. In this algorithm, data is uniformly distributed to the 8 processors, which are able to work concurrently by accessing only its respective data block in memory, synchronizing only occasionally. Rsearch and K-means follow Utility in terms of achieved speedups. In general, it can be observed that clustering algorithms show better scalability than the remainder of the applications. The underlying reason for this observation is the highly parallelizable distance calculation routine, which is common to the clustering algorithms.

The worst scalability is observed for SNP and SVM-RFE. For SVM-RFE, the problem arises due to unnecessary communication problems and locking of memory structures. This redundant locking is done to ensure the code works on distributed and shared memory machines.

For the Utility mining application, the small dataset represents real data collected from a grocery store. The large dataset has been created by the IBM Quest data generator. Both of the datasets contain a nearly equal number of transactions and items. However, the speedups for these two datasets differ widely. Particularly, the application achieves 7.55x speed-up for the small and 2.23x speed-up for the large datasets when executed on 8 processors. When the most time consuming functions are examined, it is seen that the program spends approximately 30% and 50% of the total execution time in the serial database read function, respectively. The change in the time of this serial segment causes the scalability problems for the large dataset.

Intel researchers have done similar analysis for the performance scalability of the bioinformatics workloads [13]. When the above presented results are compared to their results, Genenet, Semphy, Rsearch, and PLSA show very similar scalability trends. However the results are very different for SNP and SVM-RFE, where they are able to have close to linear speedup until 8 processors and super-linear speedup for 16 processors. The explanation given for this super-linearity is that Intel's system is composed of a 16-way shared memory machine, which has a large L3 cache and Cell-sharing L4 caches (4 processors grouped together) that are interconnected with each other through the crossbar. Specific

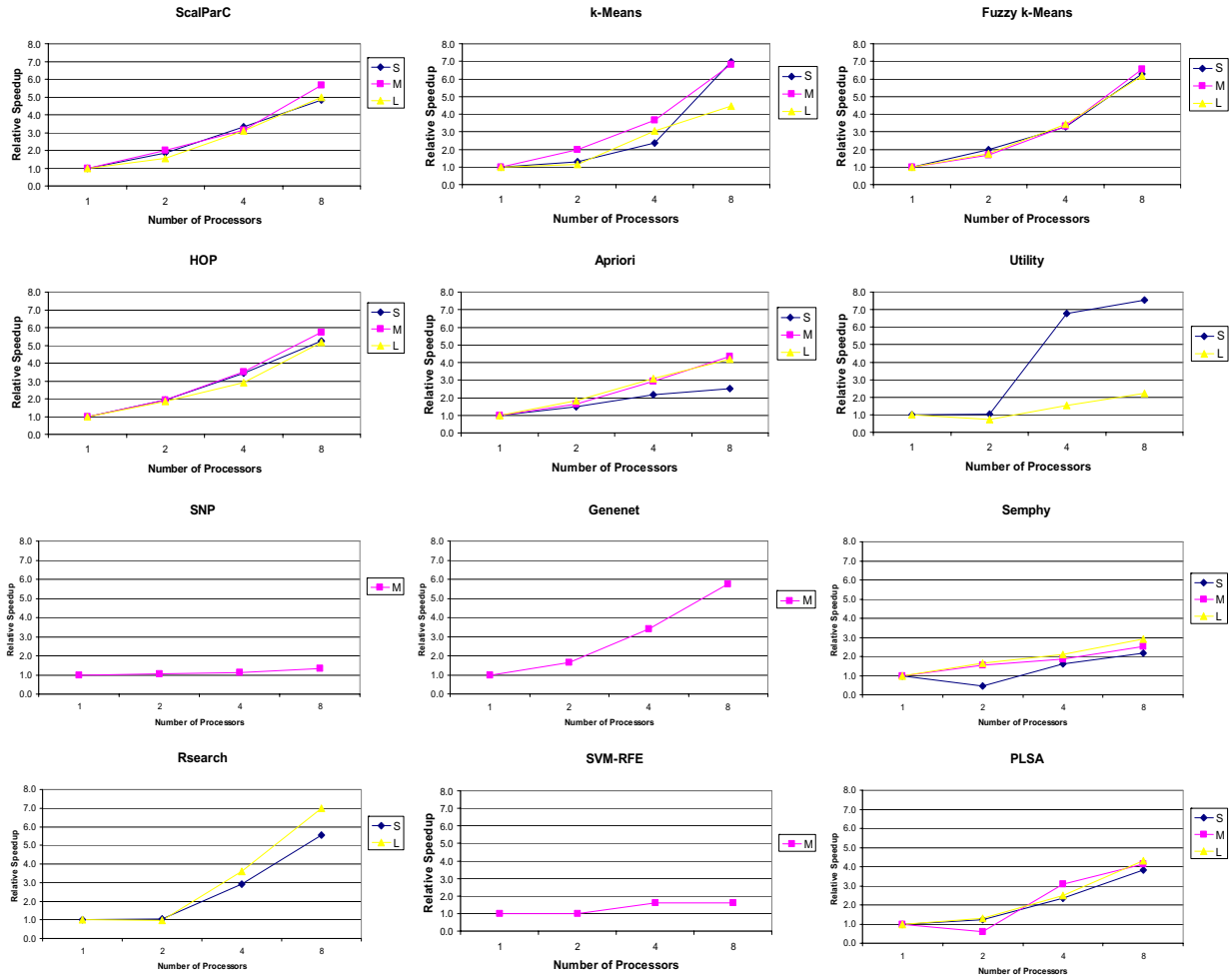

Fig. 3. Speedups for the MineBench applications

optimizations have been applied to these codes targeting their system. However, other researchers from Intel have shown that SVM-RFE reaches only 2.3x speed-up on their 4-way shared memory machine [19]. This is a sign that architecture plays an important role in the performance of these applications.

B. Memory Hierarchy Behavior

It is well known that memory hierarchy is a major performance bottleneck in modern computing systems. It is therefore necessary to understand the memory hierarchy behavior of data mining applications before attempting to improve performance. Figures 4 and 5 summarize the results obtained for memory hierarchy behavior (level 1 data, and level 2 caches, respectively) over 1, 2, 4 and 8 processor runs on medium sized datasets, wherever applicable. We notice several interesting aspects of memory behavior from these results. First, though L1 data cache miss rates are usually small, applications are drastically different in their L1 data cache behavior. We can separate the applications into two categories: those that have very small L1 data miss rates (less than 1.5%), and those that have larger miss rates (2-14%). It is also interesting to note that even in applications with low L1 data miss rates, in many cases, the 2-processor run yields much higher cache misses than the other runs. In general, we see that as the number of processors increase, L1 data cache miss rates decrease. This is due to the fact that multiple processors are working on a smaller chunk of data. Note that, for some applications, the miss rates are independent of the number of processors. In these applications, most misses are caused by cold and invalidation misses, hence they are largely unaffected by the number of processors. We also studied the L1 instruction cache miss rates. In general the L1 instruction cache miss rates are very low (on average 0.11%). This is due to the fact that the applications are relatively small in size and the instructions are able to fit into the L1 cache easily. More importantly, most of the execution in these applications are in the relatively small number of frequently executed kernels. Since the miss rates during the execution of these kernels are low, the overall instruction cache misses remain low. We have not observed much variance of instruction miss rate while going from 1 processors to 8 processors, because these applications, in general, use data parallelization concepts.

An analysis of the L2 cache behavior was also carried out

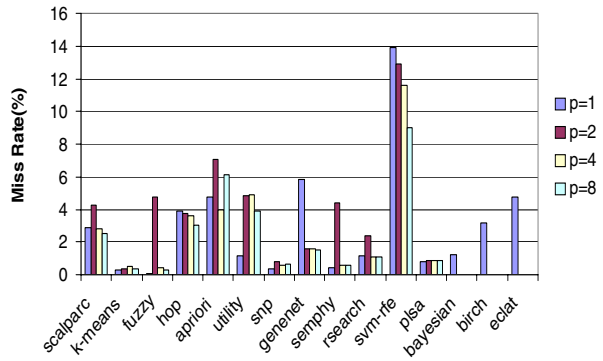

Fig. 4. L1 Data Miss Rates

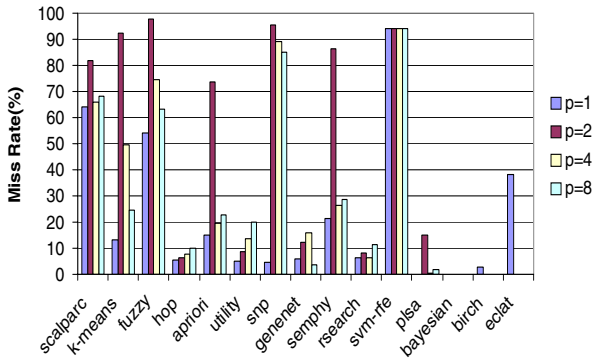

Fig. 5. L2 Cache Miss Rates

and yielded several unique characteristics. It is seen that L2 cache miss rates are many times greater than their corresponding L1 counterparts. Generally, there are two reasons for such high L2 miss rates. First, for some applications the L1 miss rates are extremely small, as a result most of the L2 accesses result in cold misses. Second, several applications work on very large datasets in a streaming fashion. Overall, the SVM-RFE benchmark had the worst L2 cache miss rates. Combined with its low L1 efficiency, approximately 8.44% of all data references incur costly off-chip memory access, thus yielding a very low IPC for this application. Another interesting observation is that in majority of the applications, the L2 miss rate for the 2 processor case is highest. One reason for this kind of behavior is that the data distribution is random as dynamic scheduling is used for parallelization in some of the applications. In dynamic schemes, the processor gets assigned a new block of data in a random fashion as it becomes available. Hence the data gets distributed to multiple caches in a random fashion, which increases the likelihood of not exploiting temporal or spatial data locality.

C. Instruction Efficiency

We also studied the instruction efficiency using the counters profiled by VTune. Particularly, we measure the branch misprediction rates, the fraction of floating-point instructions, resource related stalls (stalls caused by register renaming buffer entries, memory buffer entries, and branch misprediction recovery), and the Instructions per Cycle (IPC) values observed. These results are summarized in Figures 6, 7, 8, and 9, respectively.

In general, the branch prediction performs very well, with an average misprediction rate of 3.27% for the 15 applications. This is mostly due to the fact that the applications have small kernels which consist of loops that execute for very large number of iterations. Also, the applications are parallelized using OpenMP, which is good at analyzing large loops to extract data parallelism in an efficient way. The highest branch misprediction rate is observed for the HOP and Apriori applications. In both cases, this is partly due to the paradigm applied to parallelize the algorithms. In these two applications, the dataset is read in parallel and each processor works on local data for the most part, only synchronizing occasionally. The application does not have a concise kernel that is executed repeatedly, hence the branch misprediction increases. It is also seen that, in most applications, the branch misprediction rate decreases as the degree of parallelism increases.

We also looked at the percentage of floating point operations performed by the applications. The results are presented in Figure 7. Several of the MineBench applications are floating point intensive. As the degree of parallelism increases, it is seen that the percentage of floating point operations decreases (the number of floating point operations are usually about the same across different number of processors, but the number of instructions retired increases, thereby reducing the fraction of FP operations). Note that Apriori, GeneNet and PLSA are integer applications and do not contain any floating point operations.

Figure 8 presents the resource related stall rates for each application. It is seen that most applications suffer from high stall rates. Particularly, the SVM-RFE application spends 92% of its execution time on stalls. Since this application exhibits high L1 data and L2 cache miss rates, the instructions spend more time in the pipeline, which causes an increase in the resource related stalls. In general, we also observe a correlation between the number of floating point instructions and resource related stalls. As the fraction of floating point operations increase, the processor is able to utilize its resources better and stalls less. However, this is not true for applications like Utility mining and SVM-RFE, where other effects like large cache miss rates result in higher stall rates. As the number of processors increase, in general, the resource related stalls increase. For some applications, this causes the limitation of the scalability we observe, which is described in Section 5.1.

To express the efficiency of data mining applications, the number of Instructions per Cycle (IPC) has been studied. It can be seen that some applications suffer from very low IPCs. For example, the SVM-RFE application sees an IPC value of 0.09 with 8 processors. The reason for such low IPCs are different:

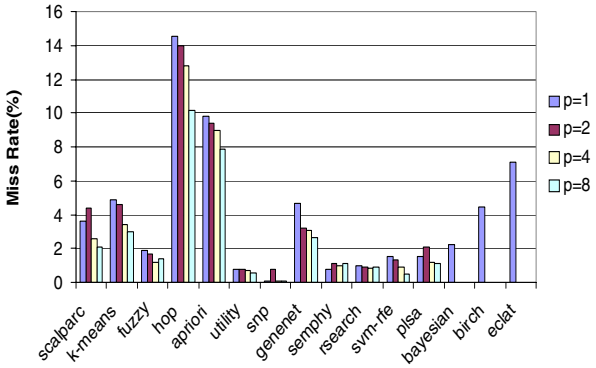

Fig. 6. Branch Misprediction Rate

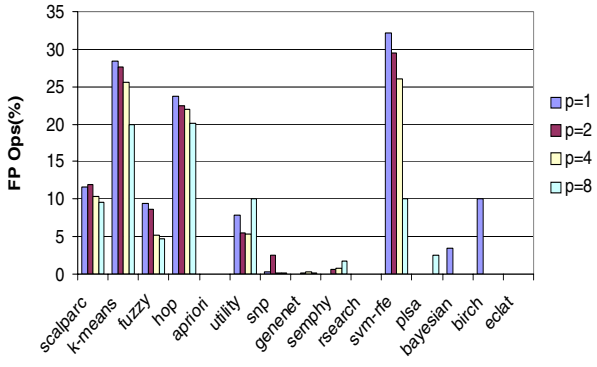

Fig. 7. Fraction of Floating Point Instructions

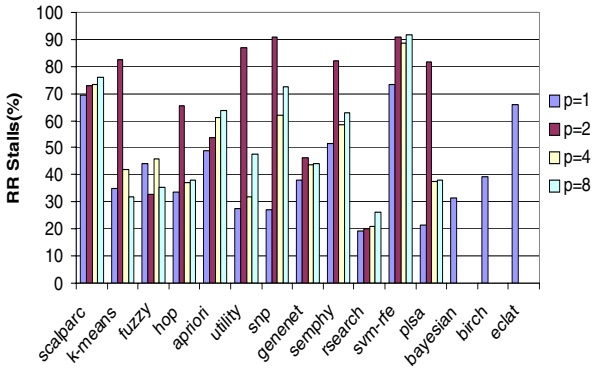

Fig. 8. Resource Related Stalls

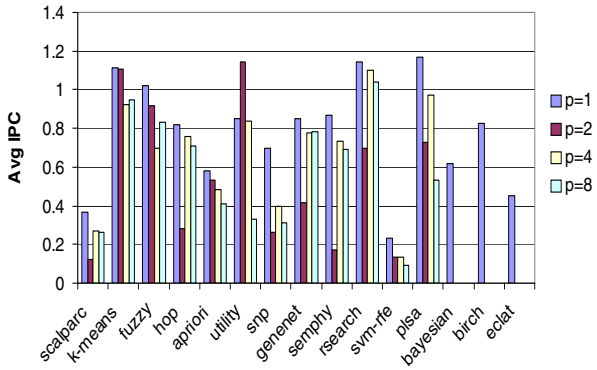

Fig. 9. Instructions Per Cycle

SVM-RFE and SNP's low IPCs are related to the high resource related stall percentages, 92% and 72% respectively; SVM-RFE, ScalparC and Utility are affected by high L1 data cache miss rates; Hop and Apriori, on the other hand, suffer from high branch mispredictions. Also, in almost all applications, as the degree of parallelism increases, the IPC decreases. In many applications, the 2-processor case experiences the worst IPC results. These results indicate that there is significant room to improve the performance of the applications by increasing their efficiencies. The parallelization of these applications also needs to be looked into, since the applications suffer from various drawbacks as the degree of parallelism increases.

VI. Conclusions

In this paper, we presented Minebench, a diverse benchmark suite of data mining applications, to enable development of superior algorithms and systems for data mining applications. Using MineBench, we establish the fact that data mining applications form a unique workload. We have studied important characteristics of the applications when executed on an 8-way SMP machine. Overall, our results indicate that there is ample scope for improvement in the performance of both data mining algorithms and systems.

To gain further insight, we have done some initial architectural simulations using an cycle-accurate x86 simulator [40]. In future, we plan to examine the workloads phase behavior to gain further insight. Also we would like to expand our benchmark with serial and parallel implimentations of similarity search, text mining and other categories that could not been included in our current version of MineBench.

MineBench is intended for use in computer architecture research, systems research, performance evaluation, and high-performance computing. MineBench is completely open and freely available for download from our Center's website [41].

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] Intel Corporation, "Architecting the era of tera - technical white paper," Available at http://www.intel.com, 2005.

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

[3] 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)*, Dec. 1997, pp. 330–335.

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

[5] L. Eeckhout, H. Vandierendonck, and K. D. Bosschere, "Quantifying the impact of input data sets on program behavior and its applications," *The Journal of Instruction-Level Parallelism*, vol. 5, pp. 1–33, Feb. 2003.

[6] 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)*, June 1995, pp. 24–36.

[7] R. Hankins, T. Diep, M. Annavaram, B. Hirano, H. Eric, H. Nueckel, and J. Shen, "Scaling and characterizing database workloads: Bridging the gap between research and practice," in *Proceedings of the 36th International Symposium on Microarchitecture*, Dec. 2003, pp. 76–87.

[8] K. Keeton, D. Patterson, Y. Q. He, R. Raphael, and W. Baker, "Performance characterization of a quad Pentium Pro SMP using OLTP workloads," in *Proceedings of the 25th International Symposium on Computer Architecture (ISCA)*, June 1998, pp. 15–26.

[9] P. Ranganathan, K. Gharachorloo, S. Adve, and L. Barroso, "Performance of database workloads on shared-memory systems with out-of-order processors," in *Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII)*, Oct. 1998, pp. 307–318.

[10] P. Trancoso, J. Larriba-Pey, Z. Zhang, and J. Torrelas, "The memory performance of DSS commercial workloads in shared-memory multiprocessors," in *Proceedings of the 3rd International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 1997, pp. 250–261.

[11] J. Bradford and J. Fortes, "Performance and memory-access characterization of data mining applications," in *Workload Characterization: Methodology and Case Studies*, Nov. 1998, pp. 49–59.

[12] J. Kim, X. Qin, and Y. Hsu, "Memory characterization of a parallel data mining workload," in *Workload Characterization: Methodology and Case Studies*, Nov. 1998, pp. 60–70.

[13] 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*, vol. 09, no. 12, pp. 131–142, May 2005.

[14] A. Jaleel, M. Mattina, and B. Jacob, "Last Level Cache (LLC) performance of data mining workloads on a CMP – a case study of parallel bioinformatics workloads," in *Proceedings of the 12th International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2006.

[15] 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)*, Sept. 2005, pp. 15–22.

[16] 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.

[17] D. Bader, V. Sachdeva, V. Agarwal, G. Goel, and A. Singh, "BioSPLASH: A sample workload for bioinformatics and computational biology for optimizing next-generation performance computer systems," University of New Mexico, Tech. Rep., May 2005.

[18] 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.

[19] U. Srinivasan, P. Chen, Q. Diao, C. Lim, E. Li, Y. Chen, R.Ju, and Y. Zhang, "Characterization and analysis of HMMER and SVM-RFE parallel bioinformatics applications," in *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, Oct. 2005.

[20] F. Sanchez, E. Salami, A. Ramirez, and M. Valero, "Parallel processing in biological sequence comparison using general purpose processors," in *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, Oct. 2005.

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

[22] 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.

[23] 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.

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

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

[26] D. Eisenstein and P. Hut, "Hop: A new group finding algorithm for N-body simulations," *Journal of Astrophysics*, no. 498, pp. 137–142, 1998.

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

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

[29] 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.

[30] M. Zaki, "Parallel and distributed association mining: A survey," *IEEE Concurrency, Special Issue on Parallel Mechanisms for Data Mining*, vol. 7, no. 4, pp. 14–25, Dec. 1999.

[31] Intel Corporation, "Intel VTune performance analyzer 7.2," Available at http://www.intel.com, 2005.

[32] 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.

[33] 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*, vol. 1, no. 4, pp. 36–47, 1999.

[34] A. Brookes, H. Lehvaslaiho, M. Siegfried, J. Boehm, Y. Yuan, C. Sarkar, P. Bork, and F. Ortigao, "HGBASE: a database of SNPs and other variations in and around human genes," *Nucleic Acids Research*, vol. 28, no. 1, pp. 356–360, Jan. 2000.

[35] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. 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*, vol. 9, no. 12, pp. 3273–3297, 1998.

[36] A. Bateman, L. Coin, R. Durbin, R. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E. Sonnhammer, D. Studholme, C. Yeats, and S. Eddy., "The Pfam protein families database," *Nucleic Acids Research*, vol. 32, no. Database, pp. D138–D141, 2004.

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

[38] C. Ambroise and G. J. McLachlan, "Selection bias in gene extraction on the basis of microarray gene-expression data," *Proceedings of the National Academy of Sciences*, vol. 99, no. 10, pp. 6562–6566, 2002.

[39] 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.

[40] R. Narayanan, B. Ozisikyilmaz, J. Zambreno, G. Memik, and A. Choudhary, "MineBench: A benchmark suite for data miningworkloads," in *To Appear in Proceedings of the International Symposium on Workload Characterization (IISWC) (Benchmark Submission)*, Oct. 2006.

[41] 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.