PERFORMANCE EVALUATION AND CHARACTERIZATION OF SCALABLE DATA MINING ALGORITHMS

Ying Liu, Jayaprakash Pisharath, Wei-keng Liao, Gokhan Memik, Alok Choudhary, Pradeep Dubey*
Department of Electrical and Computer Engineering, Northwestern University
Evanston, IL – 60208. USA.
{yingliu, jay, wkliao, memik, choudhar}@ece.northwestern.edu

ABSTRACT

Data mining has become one of the most essential tools in diverse fields. The increases in data sizes and algorithmic complexities require the computational power of chip to increase even further. In this paper, we present detailed characteristics from the hardware and software perspectives for a set of representative data mining programs. We first design MineBench, a benchmarking suite containing representative data mining applications from multiple categories including two classification, two association rule mining, and four clustering applications. We evaluate the MineBench applications on an 8-way Shared Memory Parallel (SMP) machine and analyze their important performance characteristics. During the evaluation, the input datasets and the number of processors used are varied to measure the scalability of the applications in our benchmark suite. We present the results based on characteristics such as scalability, I/O complexity, fraction of time spent in the OS mode, and communication/synchronization overheads. This information can aid designers of future systems as well as programmers of new data mining algorithms to achieve better system and algorithmic performance.

KEY WORDS

Performance evaluation, data mining, benchmark, parallel computing

1. Introduction

As the data sizes accumulated from various fields are exponentially increasing, data mining techniques that extract information from huge amount of data have become popular in commercial and scientific domains, including marketing, customer relationship management, scoring and risk management, recommendation systems, fraud detection, cosmology simulation, climate modeling, bioinformatics, drug discovery, intrusion detection, World Wide Web, etc [1].

As the amount of data collected increases, we will need to utilize even more complicated data mining applications. However, the performance of computer systems is improving at a slower rate compared to the increase in demand for data mining applications. Recent 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. 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.

In this paper, we try to investigate data mining applications to identify their characteristics in a sequential as well as a parallel execution environment. We first establish a benchmarking suite of applications, called MineBench, that encompass algorithms commonly used in data mining. We believe that analyzing the behavior of a complete benchmarking suite will certainly give a better understanding of the underlying bottlenecks for data mining applications. Such studies have been done for other fields, for instance, SPEC [2], TPC [3], SPLASH [4], MediaBench [5] are well known benchmarks built through such an effort. Then, we analyze the architectural properties of these applications, and also study their scalabilty with respect to these characteristics. We analyze the characteristics of the applications in Shared Memory Parallel (SMP) machine. Despite their limitation on scalability, SMPs have become the most common parallel computing type in industry due to their simplicity. By analyzing the application characteristics in this representative multiprocessor system, we provide an insight into the parallel applications, which can be

* From Intel Corporation.

potentially helpful when developing parallel data mining algorithms on SMPs.

The rest of the paper is organized as follows. In the next section, we overview the related work. In Section 3, we discuss the data mining applications that are included in our benchmarking suite. Section 4 presents the evaluation methodology. The characteristics of our benchmark applications are presented in Section 5. Section 6 summarizes the results.

2. Related Work

A great number of efficient data mining technologies have been developed in the recent years [1, 6]. In order to meet the demand in performance, many parallel data mining algorithms have also been developed. In the past decade, most research on parallel data mining [1, 7, 8] has been focused on Distributed Memory Parallel machines due to its capability for massive parallelism. However, Shared Memory Parallel machines are becoming the dominant types of parallel machines in industry because of its simplicity and low to medium degree of parallelism besides its nominal price. A few parallel algorithms on SMPs have been proposed in [9, 8].

Similar performance characterization work of database workloads is seen in [10, 11], and specifically targeted for SMPs in [12, 13]. Performance characterization of individual data mining algorithm has been done in [14, 15], where they focus on the memory and cache behaviors of a decision tree induction program. However, we believe that analyzing the behaviors of a complete data mining benchmarking suite will certainly give a better understanding of the underlying bottlenecks for data mining applications.

3. MineBench Application Suite

We first establish MineBench, a benchmarking suite containing data mining applications. The selection principle is to include categories and applications that are commonly used in industry and are likely to be used in the future, thereby achieving a realistic representation of the existing applications. MineBench can be used by both programmers and processor designers for efficient system design. MineBench has 8 applications from three of the categories: classification, association rule mining (ARM), and clustering. The applications as well as important characteristics of the applications are listed in Table 1, which presents the applications, the category they belong to, a short description of the applications, and the programming language used to implement it.

Table 1. MineBench applications.

Algorithms	Category	Description	Lang.
ScalParC	Classification	Decision tree classifier	C
Naïve Bayesian	Classification	Statistical classifier based on class conditional independence	C++
K-means	Clustering	Partitioning method	C
Fuzzy K-means	Clustering	Fuzzy logic based K-means	C
BIRCH	Clustering	Hierarchical method	C++
HOP	Clustering	Density-based method	C
Apriori	ARM	Horizontal database, level-wise mining based on Apriori property	C/C++
Eclat	ARM	Vertical database, break large search space into equivalence class	C++

3.1 Classification Programs

A classification algorithm is to use a training dataset to build a model such that the model can be used to assign unclassified records into one of the defined classes. Classification has applications in diverse fields such as retail marketing, fraud detection, and design of telecommunication service plans [1].

ScalParC is an efficient and scalable variation of decision tree classification [7]. The decision tree model is built 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. Among many classification methods proposed over the years, decision trees are particularly suited for data mining, since they are built relatively fast compared to other methods, obtaining similar or often better accuracy [16], and easy to interpret [17].

Bayesian classifiers are statistical classifiers based on Bayes' Theorem. They predict the probability that a record belongs to a particular class. A simple Bayesian classifier, called **Naive Bayesian** classifier [18], is comparable in performance to decision trees and exhibits high accuracy and speed when applied to large databases.

3.2 Clustering Programs

Clustering is the process of discovering the groups of similar objects from a database to characterize the underlying data distribution. It has wide applications in market or customer segmentation, pattern recognition, biological studies, and spatial data analysis [1]. Generally, clustering algorithms can be classified into four categories: partitioning-based, hierarchical-based, density-based, and grid-based.

K-means [19] is a partition-based method and is arguably the most commonly used clustering technique. 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 some similarity function. Once the assignments are completed, new centers are found by the mean of all the objects in each cluster. This process is repeated until two consecutive iterations generate the same cluster assignment.

K-means assigns a data object to be not to be a member of a particular cluster. The **Fuzzy K-means** algorithm [20] relaxes this condition by assuming that a data object can have a probability of membership in each cluster. For each object, the sum of the probabilities to all clusters equals to 1. Compared to 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 [21] is a hierarchical clustering method that employs 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 data objects.

Density-based methods grow clusters according to the density of neighboring objects or according to some other density function. **HOP** [22], 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 has diverse applications in molecular biology, geology, and astronomy.

3.3 Association Rule Mining (AMR) Programs

Association rule mining is to find the set of all subsets of items or attributes that frequently occur in database records. ARM can discover interesting association relationships among the large number of business transaction records. This can aid business decision-making processes, such as catalog design, cross-marketing, and loss-leader analysis [1].

Apriori [23] is arguably the most influential ARM algorithm. It explores the level-wise mining of Apriori property: all nonempty subsets of a frequent itemset must also be frequent. At the kth iteration (for $k > 1$), it forms frequent (k+1)-itemset candidates based on the frequent k-itemsets and scans the database to find the complete set of frequent (k+1)-itemsets, L_{k+1}. To improve the efficiency, a hash-based technique is used to reduce the size of the candidate set.

Eclat [8] 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)-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 Parallel Implementation

As mentioned in the earlier sections that part of our goal is to test the scalability of data mining applications on SMPs, parallel versions of our benchmark applications are also provided. We include sequential (serial/single processor) implementation and evaluation results for all applications. Experimental results for 5 parallel applications out of the 8 benchmark applications have been provided: ScalParC (Classification), K-means, Fuzzy K-means, HOP (Clustering), and Apriori (ARM). We chose these applications since these parallel algorithms are commonly found in the literature. ScalParC is parallelized on SMPs using the scheme presented in [9]. Simple data parallelism is exploited to parallelize K-means, Fuzzy K-means, and HOP. We implement parallel Apriori based on the Common Candidate Partitioned Database (CCPD) strategy proposed in [8].

4. Evaluation methodology

We study each application both from the algorithmic and the system perspective. Routines within each application are analyzed in detail both from the functional and architectural granularity to identify the key parameters in each algorithm.

4.1 Hardware Platform

We conduct our evaluation on an Intel IA-32 multiprocessor platform, which consists of an Intel Xeon 8-way Shared Memory Parallel (SMP) machine running Linux operating system, a 4GB shared memory and 1024 KB L2 cache for each processor. Each processor has 16KB non-blocking, integrated L1 instruction and data caches. The number of processors is varied to study the scalability.

4.2 Software Tools

In all the experiments, we use VTune Performance Analyzer [24] for profiling the functions within our application, and for measuring their breakdown execution times. VTune counter monitor provides a wide assortment of metrics. We look at different characteristics of the applications: execution time, fraction of time spent in the OS space, communication/synchronization complexity, and I/O complexity.

In parallel implementations of the applications, we use OpenMP pragmas [25]. OpenMP is a specification for a set of compiler directives, library routines, and environment variables that can be used to specify shared memory parallelism. Due to its simplicity, OpenMP is quickly becoming one of the most widely used programming styles for SMPs. In SMPs, processors communicate through shared variables in the single memory space. Synchronization is used to coordinate

processes. VTune provides the aggregate time spent on different types of pragmas, so that we can accurately measure the time spent on synchronization.

4.3 Dataset Characteristics

Input data is an integral part of the data mining applications. For ScalParC and Naïve Bayesian, we use a synthetic dataset, F26-A32-D250K, generated by the IBM Quest data generator [26]. The notation F26-A32-D250K denotes a dataset with **F**unction 26, **A**ttribute size 32, and **D**ata comprising of 250,000 records. For Apriori and Eclat, we also use a synthetic dataset, T20-I6-D2000K, from IBM Quest data generator. This notation denotes the dataset contains 2,000,000 transactions, the average transaction size is 20, and the average size of the maximal potentially large itemsets is 6. The number of items is 1000 and the number of maximal potentially large itemsets is 2000. For HOP and BIRCH, we use a real data from a cosmology application, ENZO [27], having 3,932,160 particles. We use a real image database for K-means and Fuzzy K-means. This database consists of 17,695 scenery pictures. Each picture is represented by two features: color and edge. We use the dataset represented by edge in our experiment. Since the clustering quality of K-means methods highly depends on the input parameter k, we perform both K-means with ten different k values ranging from 4 to 13. The timing results provided in this paper are the accumulated time for the ten runs.

5. Program Characteristics

In this section, we analyze several characteristics of MineBench programs. For each characteristic, we analyze how the results vary when we change the number of processors used in the execution. Our measures of interest include the overall program execution time, the operating system overheads, I/O times and synchronization times. The benefits and drawbacks of using a shared memory model for our data mining algorithms are also discussed.

5.1 Execution Time

Table 2 shows the application execution times on 1 processor and speedups with respect to 1 processor case. We measure the scalability of the parallel applications by executing them on 1, 4 and 8 processors. The performance numbers for the 2-processor case is not presented in our paper due to the fact that there is trivial (or in some cases, none) improvement in performance when it is executed on 2 processors.

The best speedup, 6.06 on 8 processors, is seen in ScalParC. The balanced data partition on to processors minimizes the memory access contention for concurrent read-write operations on the shared variables. If data is evenly distributed, each processor is able to work independently (faster) by accessing only its respective data block in the memory without requiring access to memory blocks of other processors. HOP follows ScalParC in terms of the achieved speedups. Apriori has limitations when extended to SMPs. This is due to the significant amount of atomic access to the shared hash-tree structure and the nature of unbalanced transaction data.

Table 2. Execution times for applications on 1 processor (in seconds) and speedups respect to 1 processor case. P1, P4, P8 represent 1, 4 and 8 processor cases.

Program	P1(s)	P4	P8
HOP	52.7	1.92	6.06
K-means	12.9	3.9	4.96
Fuzzy K- means	146.8	3.44	5.42
BIRCH	31.7	-	-
ScalParC	110.6	3.88	5.12
Bayesian	25.1	-	-
Apriori	102.7	2.66	3.36
Eclat	81.5	-	-

5.2 Operating System Overheads

In any program, the CPU utilization is split into operating system (OS) and user space. The OS overheads include factors like system calls (for process/thread management, invoking locks, handling hardware interrupts), and allocation of intermediate system buffers during program execution. In Figure 1, we present the OS component (as a percentage of total execution time) of each individual application. When the number of processors is 1, the operating system overheads are minimal. The maximum overhead (1.7%) on 1 processor is seen for BIRCH. When the number of processors deployed is increased, the OS component increases drastically due to the parallelization overheads. Under the OpenMP programming environment, each OpenMP (omp) directive adds extra cycles of overhead. Program locks (which are basically system locks) used in parallelization also contribute to the OS overheads. Collectively, the more processors, the more OS overheads. Among the applications, K-means has the worst overhead, 40%. It helps explain the poor scalability of K-means. This is as a result of the omp directives and locks during the parallelization of K-means.

5.3 I/O Time

In general, I/O is a key component that could affect the overall performance of a system. We study the time for performing I/O as a percentage of the overall execution time in Figure 2. It is seen that the overheads arising from I/O operations in MineBench are small except for Bayesian. For Bayesian, data is read as ASCII characters one by one, whereas for ScalParC (another classification

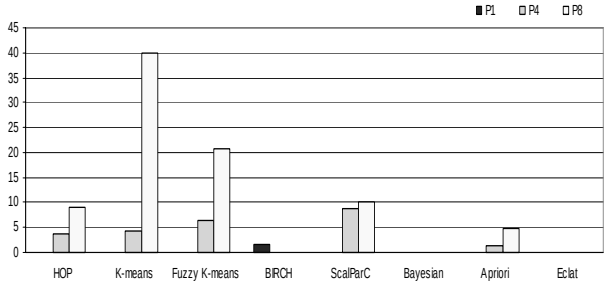

Figure 1. OS overheads of MineBench applications as a percentage of the total execution time.

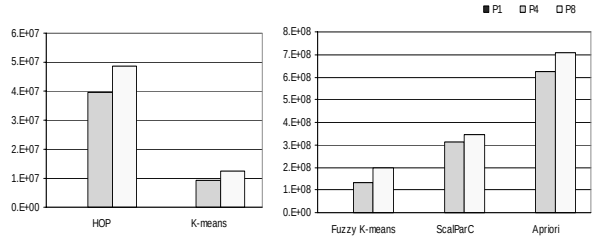

Figure 3. Synchronization time in CPU cycles for all applications. The synchronization time increases when computation is scaled to multiple processors.

algorithm), data is read in bulk string mode (less read operation overheads). These results highlight the computation-intensive nature of our benchmark.

5.4 Communication/Synchronization Overheads

In a shared memory model, the inter-processor communication is achieved by accessing shared variables (which in turn are shared locations in the memory). This could be a considerable bottleneck if the shared variable is locked by another processor, in which case the requesting processor must wait until the lock is released. Moreover, during parallel execution, there are execution breakpoints where all processors need to synchronize their data values for all their local/shared variables. This again, could be another bottleneck. All such overheads are reflected in the synchronization costs shown in Figure 3. When using one processor, the synchronization overheads are negligible due to no inter-processor communication. In our benchmark, the synchronization overheads increase when more processors participate because shared and private variables arise. It is seen that for all parallel applications, the average synchronization time is just 0.14% of the overall execution time. This implies that the idle time spent in synchronization is very less and the CPU is very well utilized for mining information from the input data.

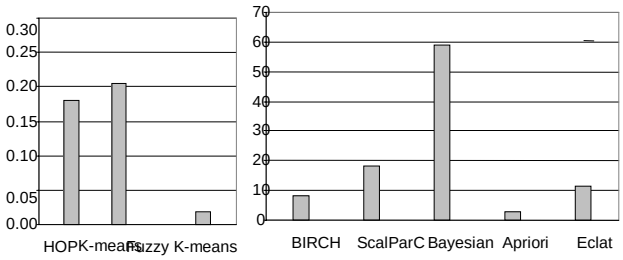

Figure 2. Percentage of I/O time with respect to the overall execution times.

6. Conclusion

In this paper, we introduce and evaluate MineBench, a benchmarking suite for data mining applications. It contains 8 representative applications: two association rule mining algorithms, two classification algorithms, and four clustering algorithms. We have studied important characteristics of the applications when executed on an 8-way SMP machine. Our results indicate that usually the OS overheads, the synchronization overheads, and the I/O time are usually small in MineBench applications. These results indicate that improvements in the performance of processors are likely to have a significant impact on the overall performance of data mining systems. In addition, techniques, like prefetching, should also improve the performance of the processor considerably. To improve the performance of their applications, system designers as well as programmers can utilize the characteristics of MineBench and achieve better system performance.

4. Acknowledgements

This work was supported in part by Intel Corporation, NSF grant CNS-0406341 and a grant from DOE's SCIDAC program.

References:

[1] Han, J. and M. Kamber, *Data Mining: Concepts and Techniques* (Morgan Kaufmanm, 2001).

[2] Standard, P.E.C., *Spec CPU2000: Performance Evaluation in the New Millennium* (Version 1.1. December 27, 2000).

[3] TPC, Transaction Processing Performance Council, *http://www.tpc.org/*.

[4] Woo, S.C., et al., The SPLASH-2 Programs: Characteriazation and methodological considerations, *International Symposium on Computer Architecture*, June 1995.

[5] Lee, C., M. Potkonjak, and W.H. Mangione-Smith, MediaBench: A Tool for Evaluating and Synthesizing

Multimedia and Communicatons Systems, *International Symposium on Microarchitecture*, 1997.
[6] Michalski, R.S., I. Brakto, and M. Kubat, *Machine Leaning and Data Mining: Methods and Applications* (John Wiley & Sons, New York, 1998).
[7] Joshi, M.V., G. Karypis, and V. Kumar, ScalParC: A New Scalable and Efficient Parallel Classification Algorithm for Mining Large Datasets, *International Parallel Processing Symposium*, 1998.
[8] Zaki, M.J., Parallel and Distributed Association Mining: A Survey, *IEEE Concurrency*, *Special Issue on Parallel Mechanisms for Data Mining*, *7*(4), Dec. 1999, 14-25.
[9] Zaki, M., C.-T. Ho, and R. Agrawal, Scalable Parallel Classification for Data Mining on Shared-Memory Multiprocessors, *IEEE International Conference on Data Engineering*, March 1999.
[10] Hankins, R., et al., Scaling and Charaterizing Database Workloads: Bridging the Gap between Research and Practice, *Intl. Symposium on Microarchitecture*, 2003.
[11] Keeton, K., et al., Performance Characterization of a Quad Pentium Pro SMP using OLTP Workloads, *Intl. Symposium on Computer Architecture*, 1998.
[12] Ranganathan, P., et al., Performance of database workloads on shared-memory system with out-of-order processors, *8th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, San Jose, CA, 1998.
[13] Trancoso, P., et al., The memory performance of DSS commercial workloads in shared-memory multiprocessors, *Third International Symposium on High-Performance Computer Architecture (HPCA)*, Jan. 1997.
[14] Bradford, J.P. and J. Fortes, Performance and Memory-Access Characterization of Data Mining Applications, *Workload Characterization: Methodology and Case Studies*, Dallas, TX, Nov. 1998.
[15] Kim, J.-S., X. Qin, and Y. Hsu, Memory Characterization of a Parallel Data Mining Workload, *Workshop on Workload Characterization: Methodology and Case Studies*, Dallas, TX, Nov. 1998.
[16] Michie, D., D.J. Spiegelhalter, and C.C. Taylor, *Machine Learning, Neural and Statistical Classification* (Ellis Horwood, 1994).
[17] Quinlan, J., *C4.5 Programs for Machine Learning* (Morgan Kaufmann, 1993).
[18] Domingos, P. and M. Pazzani, Beyond independence: Conditions for optimality of the simple Bayesian classifier, *13th Internaltional Conference on Machine Learning*, 1996.
[19] MacQueen, J, Some Methods for Classification and Analysis of Multivariate Observations, 5*th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.
[20] Bezdek, J.C., *Pattern Recognition with Fuzzy Objective Function Algorithms* (Plenum Press, New York, 1981).
[21] Zhang, T., R. Ramakrishnan, and M. Livny, BIRCH: An efficient data clustering method for very large databases, *SIGMOD*, June 1996.
[22] Eisenstein, D.J. and P. Hut, HOP: A New Group Finding Alogrithm for N-Body Simulations, *Journal of Astrophysics*, 1998, 498: pp. 137-142.
[23] Agrawal, R., et al., Fast Discovery of Association Rules, *Advances in Knowledge Discovery and Data Mining*, 1995.
[24] Intel, C., VTune Performance Analyzer, *http://www.intel.com/software/products/vtune/*.
[25] OpenMP, OpenMP: Simple, Portable, Scalable SMP Programming, *http://www.openmp.org/*.
[26] IBM, IBM synthetic data generation code, *http://www.almaden.ibm.com/software/quest/Resources/index.shtml*.
[27] Bryan, G., T. Abel, and M. Norman, Achieving Extreme Resolution in Numerical Cosmology Using Adaptive Mesh Refinement: Resolving Primordial Star Formation, *SuperComputing*, Nov. 2001.