25847
25846

Mining Protein Data Using Parallel/Distributed Association Rules

Hesyam Awadh Abdallah Bahamish, Rosalina Abdul Salam, Rosni Abdullah,
Mohd. Azam Osman, Nuraini Abdul Rashid
School of Computer Science, Universiti Sains Malaysia, 11900 Penang Malaysia.
Tel: +604-6533888 ext. 2486
E-mail: com20035@cs.usm.my , rosalina@cs.usm.mym, rosni@cs.usm.my, azam@c.usm.my, nuraini@cs.usm.my

Abstract

In biology the high-throughput technologies have led to exponential growth in genomic data, this data is flooding in at unprecedented rate. Association rules are one of the most researched data mining areas and one of the popular and widely used advanced data mining technique. They consist of two steps: finding the frequent itemsets and generating the association rules from the frequent itemsets.

The first step of association rules mining is computationally and I/O intensive. It demands a lot of CPU resources and disk I/O. Since the overall association rules mining performance is determined by this step, many sequential association rules algorithms were proposed to solve this problem. Parallel/distributed computing achieves scalability and improves the performance of compute intensive algorithms. In this research a parallel version of ITL-Mine algorithm was proposed and implemented on Distributed Memory architecture. The parallel algorithm was applied to mine the protein data extracted from SWISSPROT, PROSITE, and ENZYME databases to generate rules that concern structure, sequence and function of protein.

The performance of the parallel ITL-Mine was tested on a windows cluster of four PC machines. Results showed a significant improvement in performance by using the proposed parallel algorithm. The proposed parallel ITL-Mine algorithm can be used in other data mining task such as sequential patterns, max-patterns and frequent closed patterns, classification and clustering.

1. Introduction

The exponential growth in computer technology, scientific computing and the expanding in using computer in every life fields such as health, education, business, and industry, has resulted in more operations being computerized. All such operations accumulate data on activities and performance. The size of the accumulated data reaches to terabytes. Large databases are used to store the data. Data is continuously expanded in large scale. As an example, in biology, the high-throughput technologies have led to exponential growth in genomic data. Genomic data are flooding in at unprecedented rate. On average, the amount of information stored in protein database is doubling every 15 months [1].

The data stored in these databases is viewed as a resource, which contains much hidden valuable information, e.g. trends and patterns [2]. However, the information stored in these databases is essentially useless until analyzed [1]. There is an essential need to have a way to extract the hidden information from such huge data, and analyze it in order to help the owners of the data in getting better understanding of their data, which helps them to make right decisions.

Data mining is the extraction of previously unknown and potentially useful information from large amounts of data. It is an interdisciplinary field merging ideas from statistics, machine learning, databases, and parallel and distributed computing. It enables the analysis of huge data. Mining association rules or association rules mining in large databases is an important problem in data mining researches [4]. Association rules are one of the most researched data mining areas and one of the popular and widely used advanced data mining techniques. They are useful in deriving meaningful rules from large-scale data [4], and have recently received much attention from database community [5]. They are undirected or unsupervised data mining over variable length data and produce clear and understandable results.

Association rules were originally developed with the purpose of market-basket analysis, and were firstly introduced by Agrawal [6]. They were used to identify relationships among a set of items in a database, this relationship is based on co-occurrence of the data items [5]. Association rules have a simple problem statement which is: to find the set of all subsets of items that frequently occur in many database records and additionally, to extract rules on how a subset of items influence the presence of another subset.

0-7803-8482-2/04/$20.00 ©2004 IEEE.

Because of the surge in biological data, many challenges in biology have actually become challenges in computing [1]. There are many of data mining methods used in mining Bioinformatics data such as clustering and classification. Association rules provide the direction of the relationship unlike other methods. Since the basic framework of association rules is not dependent on the initial application problem, association rules are not limited to market basket analysis [7]. They have been shown to be useful and potentially applicable to many real world problems such as telecommunication alarm diagnosis and prediction, customer segmentation and behavior analysis, catalog design, store layout, recommender systems, diagnosis, decision support, and university enrolments [8], [9], [7].

Association rules were used in mining gene expression data [4], [10], micro array data [4], protein database [8] and protein interaction data [11]. Association rules are also computationally and I/O intensive [9]. Association rules demand a lot of CPU resources and disk I/O to solve, this especially appears in the first step of association rules mining. Finding frequent itemsets plays an essential role in many data mining tasks, such as sequential patterns, max-patterns and frequent closed patterns, classification and clustering [12]. Since the overall association rules mining performance is determined by this step, many sequential association rules algorithms were proposed to solve this problem. These algorithms can be classified into two approaches; Candidate generation and test, and pattern growth approach. Some of the sequential algorithms suffer from the poor performance when the database is large in both the size and dimension, and they lack scalability. Parallel/distributed computing achieves scalability and improves the performance of compute intensive algorithms. Parallel/distributed association rules algorithms are more scalable and have better performance in mining huge databases than sequential association rules algorithms. A few works have been done in parallel/distributed Frequent Pattern Growth algorithms.

The aim of this research is to improve the performance of ITL-Mine algorithm proposed by Gopalan and Sucahyo [13] by parallelizing it on Distributed Memory (shared nothing) architecture and using it to mine the protein data extracted from SWISSPROT, PROSITE, and ENZYME databases to generate rules that concern structure, sequence and function of protein. The second aim is to investigate the use of association rules in mining Bioinformatics data.

2. Motivation

The mining processes of association rules consist of two steps, finding the frequent itemsets and then generating the association rules from these frequent itemsets. The first step is the main step and the most intensive computing and I/O. As the overall performance of the algorithm is depended on it, the research has been focused on how to develop efficient algorithms. Parallel/distributed association algorithms are mostly proposed to improve the performance of the algorithm in finding frequent itemsets by parallelizing the processes finding frequent itemsets. The algorithms of finding frequent itemsets can be classified into two approaches the candidate generation and test approach and the frequent pattern growth approach. Sequential and parallel/distributed association rules algorithms are based on these two approaches. The second step is much less expensive than the first step and does not require scanning the database.

2.1 Sequential frequent itemsets finding algorithms

Many sequential algorithms have been proposed for finding frequent itemsets. Apriori algorithm and its several variations are based on candidate generation and test approach, this approach is also called Apriori-like approach [6]. FP-Growth [12], Pattern Repository (PR) [14], H-Mine [16] and ITL-Mine [13] are based on frequent pattern growth approach.

Using Apriori property reduces the size of candidate sets, but in large databases the number of candidates is still large. This makes the algorithm to run very slowly on long pattern data sets [13]. The performance of FP-Growth algorithm is sensitive to support threshold. FP-tree may exceed available memory space even when the data is compressed. Although the mining processes are much less costly than candidate generation and pattern matching, it requires huge space to serve the mining and the complex data structure representation makes the processes of mining expensive and computationally exhaustive.

H-Mine algorithm was proposed by Pei [16] to improve the FP-Growth algorithm. A new data structure, H-struct, and a new mining algorithm H-mine, are proposed to overcome some of the disadvantages of FP-Growth algorithm. H-mine is more efficient space preserving than FP-growth algorithm. It moderates the memory usage and it can fully utilize all available main memory space. Gopalan and Sucahyo [13] proposed a new data structure Item-Trans Link (ITL) to represent transaction data. They also proposed ITL-Mine algorithm to mine this structure to generate the frequent itemsets. ITL structure combines the horizontal and vertical data layout.

2.2 Association Rules in Mining Bioinformatics data

Association rules algorithms were used to mine Bioinformatics data. It was firstly used by Satou [8] to find rules describing the association among

heterogeneous genome data such as sequence, structures and functions of the proteins. The genome data were extracted from various genome databases. In order to apply the association rules mining algorithms on protein data, the protein data should be extracted from their databases and pre-processed and transformed into a form that can be used by the mining algorithms. Satou [8] extracted the protein data from three databases, PDB (Protein DataBase), SWISPROT, and PROSITE, and characterized it from sequential, structural, and functional viewpoints. For this characterization, they chose four data sources, which are related to each other by using PDB entry names as keys. These four data sources are: sequential feature, similar substructures as a structural feature, EC number as function, and SWISSPROT keyword as function.

After building the relationship between domains and proteins, they applied the Apriori and AprioriTiD sequential algorithms to find the correlations of domain sharing in proteins. The mining results were a set of correlations, each correlation represents a fact that a group of domains usually appear together in a set of proteins. Besides the correlations of the motifs in proteins, they also found the proteins that satisfy the correlations. They showed that many proteins share more than one motif and such motif correlation suggests that some biological function could be coupled and this coupling should shed new light on biological mechanisms and pathways [17].

Oyama [11] used association rules to detect association rules related to protein protein interaction from accumulated protein protein interaction data. The data that they used was an interaction data represented as a pair of two proteins that directly binds each other. Proteins pairs were obtained from four sources, YPD (Yeast Proteome Database), MIPS (Munich Information Center for Protein Sequences), Two-hybrid [11]. Then they classified the various protein functions and characteristics which called "features" according to several genome databases from functional, primary structural and other various viewpoints, into seven types: YDP categories, EC numbers, SWISSPROT/PIR keywords, PROSITE motifs, Bias of amino acids, Segment clusters, and Amino acids patterns.

In contrast of other applications of association rules mining algorithms in mining protein data which regarded the protein as a transaction such as [8], [17], [11] used Interaction-based representation. They found that, using the Protein-based representation in the case of mining association rules of protein interaction data was merely represented the relations between the features of a single protein, not those between the features of the two proteins appearing in the interaction. They regarded the interaction of the proteins as a transaction. Based on this representation, each transaction represents an interaction consists of two proteins.

Conklin [10] presented a new association rules mining method for discovering nucleotide sequence patterns that appear in more sequences than expected within protein function classes. They applied the method to a database of human 3' UTR sequences and found some significant associations between nucleotide patterns and protein function classes.

2.3 Parallel/distributed association algorithms

As the task of finding frequent itemsets requires a lot of computation power, it is difficult for a single processor to provide reasonable response time. Parallel and distributed computers provide high computational power and provide increased memory to solve this problem [3]. Most of the current parallel and distributed algorithms are based on their sequential versions, and most of them are based on Apriori-like approach or candidate generation and test approach [15]. A few works have been done in parallelizing the frequent pattern growth approach algorithms.

Zaki [9] classified the Parallel/distributed association rules algorithms by load-balancing strategy, architecture and parallelism. In parallelism he classified the algorithms into two paradigms based on candidate set. So this classification is for the Apriori-like algorithms. The paradigms are the data parallelism and task parallelism. The differences between the two paradigms are based on the candidate set, whether it is partitioned and distributed across the processors or not.

Zaiane [7] proposed Multiple Local Frequent Pattern Tree (MLFPT). It is a parallel algorithm based on FP-Growth algorithm, implemented on shared memory architecture. It consists of two phases. The first phase is the construction of the parallel frequent pattern trees (one for each processor) and the second phase is the actual mining of these data structures, it is much like FP-Growth algorithm.

3. Methodology

As mentioned earlier, association rules consist of two steps. In order to mine the protein data, the first step must be performed on the data to find the frequent itemsets and then the second step generates the rules from frequent itemsets. The proposed Parallel ITL-Mine algorithm is an algorithm for finding the frequent itemsets.

3.1 Data extracting and preprocessing

As the origin of the association rules mining is the market-basket analysis, the data for mining is represented as transactions contain items. To apply the association rules algorithm on protein data, the protein data has to be represented as transactions contain items. The sequence, structure features and

functions of the protein are represented as the items of the transactions. In this research, this representation is used to represent protein data. In order to prepare the data in this representation, the following fields need to be extracted from databases. These fields are:

- The proteins identifier ID and their KEYWORDS from SWISSPROT database. The KEYWORDS of protein include functional, structural characterization.
- The proteins ID and their EC number from SWISSPROT and ENZYME databases. The EC number represents the classification of Enzymes, which based on their functions, in four levels of hierarchy.
- The protein ID and PROSITE identifier ID from SWISSPROT and PROSITE databases. The sequential feature of the protein is determined by the PROSITE sites and patterns.

The SRS 7.0.2 query server from EMBL-EBI is used to extract these data fields. All the extracted data is saved in text files and imported into Microsoft Access 2002 tables to make further processing to the data to be suitable for the mining algorithms.

By using Microsoft Access 2002 query, the data is selected from the three tables and the total table is generated which contains the fields from the three tables. These fields are: ID which is considered as transaction ID. The SWISS identifier. Ten fields for protein KEYWORDS, named key1 to key10. (The protein may have 0 to 10 KEYWORDS). Five fields for the protein's PROSITE ID, named pro1 to pro5. (The protein may have 0 to 5 PROSITE identifiers). Two fields for the EC number of the protein, named enz1 and enz2. (The protein may have 0 to 2 EC numbers). As in RP and ITL-Mine Algorithms the items are mapped into integer values. To do this all items in the database are mapped into integer values and saved in a text file. Then each transaction in the total table is read and each item is converted to its integer value and saved in a text file which is the final data for mining.

3.2 Parallel ITL-Mine algorithm

A Parallel ITL-Mine algorithm tries to improve the performance of the sequential ITL-Mine algorithm by applying it on Distributed Memory (shared nothing) architecture to mine protein data. Work is distributed over the processors.

The main idea of sequential ITL-Mine algorithm is to map the database transactions into the ITL structure (ItemTable and TransLink), find the frequent 1-itemsets, prune the ITL structure and then do the mining processes on this structure starting with the frequent 1-items in the ItemTable, constructs the TempList for the item and finds the frequent itemsets. In the sequential ITL-Mine the step of pruning ITL structure costs time especially when the minimum support is high. This is because the algorithm constructs the ItemTable and TransLink for all the items in the transactions then depending on the minimum support value it prunes the ItemTable and TransLink. To over come this time cost, the algorithm was modified to construct the ItemTable first, prune the ItemTable to delete the infrequent items and then construct the TransLink only for the items in the pruned ItemTable.

The parallelism of the association rules algorithms is classified into two paradigms, data and task parallelism [9]. In data parallelism paradigm data is partitioned among p processors, each processor performs the same work on local partition. In task parallelism each processor performs different computation data may be (selectively) replicated or partitioned.

ITL-Mine algorithm can be parallelized by partitioning the database into n partitions and distributing them over p processors, where $n=p$. Among the n processors, one processor is the master processor (processor 1). Each processor constructs the local ItemTable and sends it to the master processor. The master processor combines the local ItemTables into a global ItemTable and deletes the infrequent items. The global ItemTable contains all the frequent 1-itemsets in the data. The master processor broadcasts the global ItemTable to the slave processors. The parallel ITL-Mine algorithm steps are below:

Step 1: Each processor constructs its local ItemTable. Items in the local ItemTable are sorted in ascending order. Then each processor sends its local ItemTable to the master processor.

Step 2: The master processor receives the local ItemTables. It builds a global ItemTable from the local ItemTabels. The infrequent items from the global ItemTable are deleted. Prune the local ItemTable. Broadcast the global ItemTable to the slaves processors.

Step 3: Each processor receives the global ItemTable and performs the task distribution step.

It prunes the local ItemTable. Constructs the TransLink and constructs the TempLists. Then, it sends the TempLists to their responsible processors.

Step 4: Each processor receives its TempLists from other processors. It combines them starts mining to generate the frequent itemsets. The generated itemsets are sent to the master processor.

Step 5: The master processor receives the frequent itemsets and combines them in a global frequent itemsets.

The step of generating association rules from the frequent itemsets is straightforward. Given a frequent itemset l, rule generation examines each non-empty subsets a and generates the rule $a \Longrightarrow (l\text{-}a)$ with support=support(l) and confidence=support(l) / support(a). The rules the have the predefined support and confidence will be considered. The items in the generated rules are in the integer values, to give a meaning to the generated rules; items must be converted back into their real value.

4.0 Implementation

Data was extracted and transformed into integer values. The transformation of data into integer values was done by a program written in Microsoft Visual Basic 6.0 .

4.1 Sequential and Parallel ITL-Mine

In order to evaluate and compare the performance between the parallel ITL-Mine and sequential ITL-Mine algorithms, the sequential-ITL algorithm was implemented first by using Microsoft Visual C++ 6.0 on a 2.4 GHz Pentium 4 PC machine with 256 MB main memory, 30 GB hard disk. The PC machine runs windows^{xp} professional operating system.

Based on the sequential ITL-Mine algorithm, the parallel ITL-Mine algorithm was implemented on a shared distributed architecture by using Microsoft Visual C++ 6.0 programming language. A windows cluster of four PC machines with the same specifications of the machine that used in the implementation of the sequential ITL-Mine was used for the implementation of the parallel ITL-Mine algorithm. The four PCs in the cluster are connected by a 10-100 MB LAN. The setting and configuration of the cluster were done using the MPICH configuration tool.

The final data file was partitioned into a number equal to the number of the processors. Each part was saved in the local disk for the PC machine before the running of the program. The master slave parallel architecture was used in the implementation of the parallel ITL-Mine algorithm. All the processors run the same program. The Master processor works as a slave and master at the same time. It has specific jobs to perform such as combining the local ItemTable, broadcasting the global ItemTable, combining the frequent itemsets, generating the rules and converting the items in the rules from the integer values to the original names of the items. Besides the master specific jobs the master processor mines its data part to find the frequent itemsets.

On the execution of the parallel ITL-Mine, The program in the master processor prompts the user to enter the value of the two parameters the minimum support and the minimum confidence. The master processor broadcasts the minimum support value to all slave processors. The master processor does not broadcast the minimum confidence value to all the slave processors because the minimum confidence value will not be used in the slave processors. The minimum confidence value is used only in the master processor during the step of rules generation.

All processors start to construct the local ItemTable and send the number of transactions found in their data parts to the master in order to accumulate the total number of transactions in all the data parts. The master then broadcasts the total transactions number to all the slave processors. Having the total transactions number is needed for each processor in order to calculate the support value of each itemsets.

Slave processors send their local ItemTables to the master processors. Master processor combines the local ItemTable, builds the global ItemTable, prunes the global ItemTable, and broadcasts the global ItemTable to all the slaves. In order to distribute the work equally between the processors, each processor performs the task distribution function. The task distribution function simply divides the number of the frequent 1-itemsts in the global ItemTable into equal parts. The numbers of the parts are equal to the number of the processors. The function then assigns each part to a processor. So each processor is responsible for a part of the items in the global ItemTable. Now, each processor works independently to construct the TransLink and the TempLists.

After the construction of the Templists, the exchange of the TempLists between processors is needed in order to generate all the frequent itemsets. Each processor receives the TempLists of its items from other processors and sends the TempLists of the other processors items to their processors. Each processor combines the received Templists, mines the Templists and finds all the frequent itemsets of its part.

Slave processors send the generated itemsets to the master processors. The master processor combines the all frequent itemsets, generates the rules that depend on the minimum confidence and support values, converts the items in the rules into their original names and save them in a text file.

5. Experimental Results

Parallel ITL-Mine was proposed to improve the performance of the sequential ITL-Mine algorithm. The parallel ITL-Mine algorithm was tested with 2, 3 and 4 processors using a minimum support value ranged from 1% to 16% and minimum confidence 100%. The results were compared with the results of the sequential ITL-Mine. A ratio of 39.29%, 57.20% and 61.69% of execution time reduction of the sequential ITL-Mine execution time were achieved when 2, 3 and 4 processors were used receptively.

Figure 1 shows the reduction of execution time when the number of processors increased. The parallel ITL-Mine algorithm achieved the reduction of the execution time because of the distribution of the data over the processors where

each processor worked on its data to construct the ItemTable, TransLink and TempList and communicate with other processors to complete its work. The distribution of the data reduced the computation time for constructing the ItemTable, TransLink and TempList which reduced the total execution time.

The ratio of execution time reduction when the number of processor was 3 is 29.50% of the execution time when the number of processors was 2. This ratio decreased to 7% of the execution time of 3 processors when the number of processors was 4. The reason for this small ratio of execution time reduction between 3 and 4 processors was the communication overhead which rose when the number of processors increased. The communication overhead was due to the increasing of the number of messages and data movement between processors.

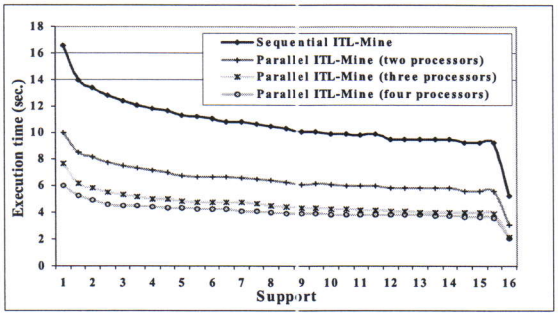

Figure 1. Parallel and sequential ITL-Mine performance execution time

6. Conclusion and Future Research

In this work the use of association rules in mining Bioinformatics data was investigated and a parallel ITL-Mine algorithm was proposed and implemented on a shared nothing parallel architecture and tested on a windows cluster of four PC machines. Data for mining was extracted from three databases (SWISSPROT, PROSITE and ENZYME), preprocessed and mined by the proposed algorithm. Parallel ITL-Mine algorithm achieved a performance improvement over the sequential ITL-Mine algorithm with 61.69% execution time reduction.

Results of this work showed that the proposed parallel algorithm has superior performance on mining of association rules in Distributed Memory (shared nothing) architecture. The proposed parallel algorithm can be used in other data mining task such sequential patterns, max-patterns and frequent closed patterns, classification and clustering.

7. References

[1] Luscombe, N., Greenbaum, D. & Gerstein, M. (2001). What is Bioinformatics? An Introduction and Overview.

[2] Goebel, M. & Gruenwald, L. (1999). A survey of Data Mining and Knowledge Discovery Software Tools, *SIGKDD Exploration, ACM SIGKDD*, (1)1, pp. 20-33.

[3] Shintani, T. & Kitsuregawa, M. (1996). Hash Based Parallel Algorithms for Mining Association Rules. *In Proc. of 4th Int. Conf. on Parallel and Distributed Information Systems.*

[4] Kotala P., Perera A., Kai Zhaou J., Mudivarthy S., Perrizo W. & Deckard E. (2001). Gene Expression Profiling of DNA Microarray Data using Peano Count Trees (P-trees), http://midas-10.cs.ndsu.nodak.edu/bio/

[5] Dunham, M., Xiao, Y., Gruenwald, L & Hossain, Z. (2000). A Survey of Association Rules.

[6] Agrawal, R., Imielinski, T. & Swami, A. (1993). Mining Association Rules between Sets of Items in Large Databases. *Proc. ACM SIGMOD*, pp. 207-216.

[7] Zaiane, O, El-Hajj, M. & Lu, P. (2001). Fast Parallel Association Rule Mining without Candidacy Generation. *Technical Report TR01-12, Department of Computing Science, University of Alberta, Canada.*

[8] Satou,K., Shibayama,G., Ono,T., Yamamura,Y., Furuichi,E., Kuhara,S., & Takagi,T. (1997). Finding association rules on heterogeneous genome data. *Proceeding of the pacific symposium on biocomputing.* pp. 397-408.

[9] Zaki, M. (1999). Parallel And Distributed Association Mining: A survey. *IEEE* **(17)4**, pp. 14-25.

[10] Conklin, D., Jonassen, I., Aasland, R. & Taylor, W. (2002). Association Of Nucleotide Patterns With Gene Function Classes: Application to human 3' untranslated sequences. *BIOINFORMATICS* **,18(1),** pp. 182-189.

[11] Oyama T., Kitano K., Satou K. & Ito, T. (2002). Extraction of Knowledge on Protein-Protein Interaction by Association Rule Discovery. *BIOINFORMATICS*, 18(5), pp. 705-714.

[12] Han, J., Pei J. & Yin Y. (2000). Mining Frequent Patterns without Candidates Generation. *Proc. ACM-SIGMOD Int. Conf. On Management of Data (SIGMOD'00),* Dallas, TX, May 2000.

[13] Gopalan, R. & Sucahyo Yudho, G. (2002). ITL-MINE: Mining Frequent Itemsets More Efficiently. *FSKD 2002*, pp. 167-171.

[14] Relue, R., Wu, X. & Huang, H. (2001). Efficient Runtime Generation of Association Rules. CIKM 200, 466-473.

[15] Relue, R. & Wu, X. (2002). Rule Generation with the Pattern Repository. *IEEE International Conference on Artificial Intelligence Systems (ICAIS'02).*

[16] Pei, J., Han, J., Lu H., Nishio, S., Tang, S. & Yang, D. (2001). H-Mine: Hyper-structure Mining of

frequent patterns in large databases. *ICDM 2001*, pp. 441-448.
[17] Horng, J., Huang, H., Wang, S., Lin, F., Liu, B. & Hwang, J. (2002). Study of Motif Correlation in Proteins by Data Mining. *International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS '02)*, pp. 345-350.