Static Load Balancing of Parallel Mining Efficient Algorithm with PBEC in Frequent Sequences Dataset

Mr.Suraj Patil[1], Prof: Parth Sagar[2]

[1]P. G Student, Department of Computer Engineering, RMD Sinhgad School of Engineering, Pune, India
[2]Assistant Professor, Department of Computer Engineering, RMD Sinhgad School of Engineering, Pune, India

---***---

Abstract- *Discovery of Frequent sequence mining is an essential data mining task with broad applications .The output of the algorithm is used in many other areas like market basket analysis, chemistry and bioinformatics. The frequent sequence mining is computationally high expensive. Further sequential patterns mining in a single dimension mining and multidimensional sequential patterns can give us more constructive and useful patterns. Due to the huge enhance in data volume and also fairly large search space, efficient solutions for finding patterns in multidimensional sequence data are currently very important. For this reason, developing a frequent sequence mining algorithm is necessary. Parallel algorithm follows the step by step approach and all participating processors or workers generate candidate sequence and count their estimate time and supports independently.*

Key Words: **Data mining, frequent sequence mining, parallel algorithms, static load balancing, probabilistic algorithms.**

1. INTRODUCTION

Repeated pattern removal is an important data mining technique with a wide variety of mined patterns. The mined frequent patterns can be sets of items (item sets), sequences, graphs, trees, etc. Frequent sequence mining was first described in. The GSP algorithm presented in is the first to solve the problem of frequent sequence mining. As the repeated series removal is an extension of item set mining, the GSP algorithm is an extension of the Apriori algorithm. As a consequence of the slowness and memory consumption of algorithms described in other algorithms were proposed. These two algorithms use the so-called prefix-based equivalence classes (PBECs in short), i.e., represent the pattern as a string and partition the set of all patterns into disjoint sets using prefixes. There are two kinds of parallel computers: shared memory technology and distributed memory technology. Parallelizing on the shared memory technology is easier than parallelizing on distributed memory technology. Sampling technique that statically load-balance the computation of parallel frequent item set mining process, are proposed in these three papers, the so-called double sampling process and its three variants were proposed.

There are other problems with static load-balancing the estimation of the running time in a PBEC is a non trivial task. The intuition behind this is that exact computation of number of frequent sequences in a PBEC is at least #P complete task. The hardness also comes from the fact that the amount of work necessary to process one sequence vary among sequences. The last problem, according to our experiments, is that each processor gets almost the whole database. A method of static load-balancing, called selective sampling is presented in parallel mining. The selective sampling process estimates the running time in each PBEC by removing some items from the database.

In this paper we have Static Load Balancing of Parallel Mining Efficient Algorithm methods. Section 1 of Introduction Section 2 this paper deals with related work and Section 3 Proposed System 4 Algorithms 5 Results and Discussion Section 6 Conclusion and future work of the paper.

2. RELATED WORK

In the Static load balancing important things are view of load, assessment of load, constancy of different system, performance of system, interaction between the records , natural world of work to be transferred, selecting of data sets and many other ones to consider while developing such algorithm

In this paper,[1] Sequential Pattern Mining from Multidimensional Sequence Data in Parallel finding patterns in multidimensional sequence data are nowadays very important present a multidimensional sequence model and a parallel algorithm follows the level-wise approach and all participating processors or workers generate candidate sequence and count their supports independently.

In this paper, [2]Prefix Span Mining Sequential Patterns by Prefix Projected Pattern which discovers frequent sub sequences as patterns in a sequence database, is an important data mining problem with broad applications, including the analysis of customer purchase patterns or Web access patterns, the analysis of sequencing or time related processes such as scientific experiments, natural disasters, and disease treatments, the analysis of DNA sequences etc.

In the paper,[3] Parallel Sequence Mining on Shared-Memory Machines a parallel al-gorithm for fast discovery of frequent sequences in large databases. pSPADE decomposes the original search space into smaller sufix based classes. Each class can be solved in main memory using efficient search techniques, and simple join operations. Further each class can be solved in-dependently on each processor requiring no synchronization

In this paper, [4] Sequential mining patterns and algorithms analysis Sequential pattern is a set of item sets structured in sequence database which occurs sequentially with a specific order. A sequence database is a set of ordered elements or events, stored with or without a concrete notion of time The most used measures used to evaluate sequential patterns are the support and confidence.

In this paper,[5] Model for Load Balancing on Processors in Parallel Mining of Frequent Item sets. This market basket data consists of transactions made by each customer. Each transaction contains items bought by the customer. The goal is to see if the occurrence of certain items in a transaction can be used to deduce occurrence of other items or in other words, to find associative relationships between items

In this paper,[6] Probabilistic static load-balancing of parallel Mining of repeated series in this project we present a novel parallel algorithm for removal of repeated series based on a static load-balancing. The static load balancing is done by measuring the computational time they are slow and needs much more memory, compared to DFS algorithms.

In this paper,[7] Parallel Mining of Closed Sequential Patterns to make sequential pat- tern mining practical for large data sets, the mining process must be efficient, scalable, and have a short response time. Moreover, since sequential pattern mining requires iterative scans of the sequence dataset with various data relationship and analysis operations, it is computationally intensive.

3. PROPOSED SYSTEM

Proposed method is a novel parallel method that statically load balance the computation. The set of all frequent sequences is first split into PBECs, the relative execution time of each PBEC is estimated and finally the PBECs algorithm is assigned to processors. The method estimates the processing time of one PBEC by the sequential Prefix span algorithm using sampling data sets. It is important to be aware that the running time of the parallel sequential algorithm scales with points

1) the database size 2) the number of frequent sequences 3) the number of embeddings of a frequent sequence in database transactions.

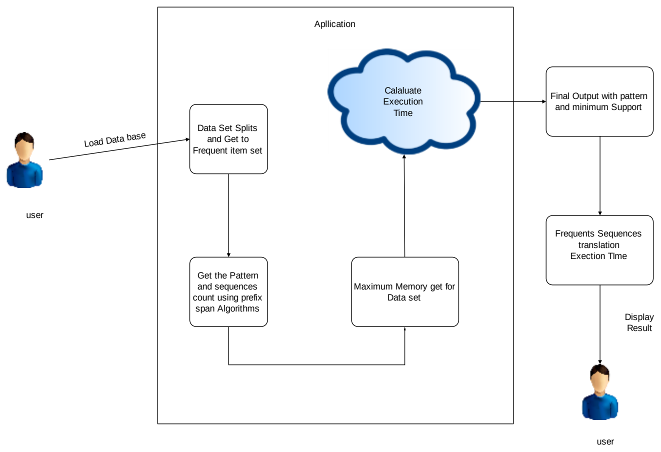
Apllication
Load Data base
user
Data Set Splits and Get to Frequent item set
Calaluate Execution Time
Final Output with pattern and minimum Support
Get the Pattern and sequences count using prefix span Algorithms
Maximum Memory get for Data set
Frequents Sequences translation Exection TIme
Display Result
user

Fig 1. Proposed System Architecture

Static load balancing of the computation begins with partitioning the set of all frequent sequences into disjoint tasks. Because the PBECs are disjoint, they perfectly fit the needs of the algorithm. the total processing time of the sequential Prefix span algorithm. The processing time of each PBEC should be evaluated. The algorithm begins splitting the set of all frequent sequences into smaller pieces recursively using PBECs. The relative size of a PBEC is the estimate of the fraction of the total processing time of a PBEC.

3.1 Module Description

1. **Estimation of Support Module**

In this module, we can estimate whether a support of sequence is a subsequence of a transaction in a database or not.

2. **Estimation of Relative Size of PBEC Module**

The relative size of a PBEC can be used as the estimate of the relative processing time of the PBEC by a sequential algorithm. This estimate ignores some details of the sequential algorithm. The relative size of PBEC might be controllable size and smaller data set.

3.2 Prefix span Algorithm: Step wise considerations

1) **Create initial collection of frequent extensions:** The algorithm starts adding items into a set of extensions S. Adding items into S is not straightforward because the collection of items requires

2) Construction of the initial pseudo projected database: Create the initial pseudo projected transaction for e for each transaction. The projection adds new event containing single item e into S.

3) Projection using a sequence extension:

Store the positions of data set into a new pseudo projected transaction that is stored in new pseudo database.

4) The projection using an event extension:

Search for the item e until the end of the event is found.

4 Algorithms:

Algorithm 1. Prefix span Sequential

```
• PREFIXSPAN SEQUENTIAL (Database D, Integer min supp)
1. Σ ← all frequent extensions from D
2. for for all e Є Σ do
3.     Q ← <(e)>
4.     create projection D|q
5.     PREFIXSPAN-MAINLOOP(D,Q,D|Q,min supp)
6. end for
```

5 RESULTS AND DISCUSSION

For the proposed system performance evaluation, we calculate frequency. We implement the scheme on Java framework with Intel Core i3 processor and 2 GB RAM. Here the graph in Fig-2 demonstrates the system performance.

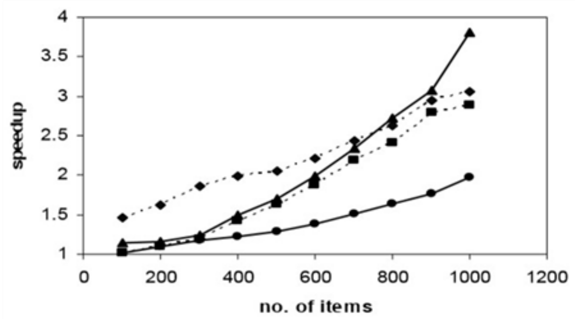

Fig.2 No of item sets

The PBECs are created, scheduled, and executed on the processors. Because the PBECs are scheduled once, we talk about static load balance of the computation. The sequential algorithm runs for too long there is a need for parallel algorithms. There is a very natural opportunity to parallelize an arbitrary frequent sequence mining algorithm partition these to all frequent sequences using the PBECs.

6 CONCLUSION AND FUTURE SCOPE

The frequent sequences and use this sample for estimating the relative processing time of the algorithm in the PBECs with evolutionary optimization technique The estimate of the relative processing time is in fact performed by estimating the computational complexity of processing various PBECs. The relative processing time is then used for partitioning and scheduling of the PBECs an algorithm for mining of frequent sequences using static load-balancing. The method creates a sample of frequent sequences the relative processing time is then used for partitioning and scheduling of the PBECs. The problem is that the estimated size of a PBEC is dependent on the construction of the PBEC (which should not happen). This dependency could be probably removed by using, for example, the bootstrap method.

In future we have to implement the parallel algorithm to reduce the complexity of computational time and implement the result of frequent sequence mining using static load balancing. Additionally, we have to reduce the slowness and memory consumption of a process.

ACKNOWLEDGEMENT

It is my privilege to acknowledge with deep sense of gratitude to my guide Prof. Parth Sagar for her kind cooperation, valuable suggestions and capable guidance and timely help given to me in completion of my paper. I express my gratitude to Prof. Vina M. Lomte, Head of Department, RMDSSOE (Computer Dept.) for her constant encouragement, suggestions, help and cooperation.

REFERENCES

[1] J. Ren, Y. Dong, and H. He, A parallel algorithm based on prefix tree for sequence pattern mining, in Proc. 1st ACIS Int. Symp. Cryptography Netw. Security, Data Mining Knowl. Discovery, E-Commerce Appl. Embedded Syst., 2010, pp. 611

[2] R. Kessl and P. Tvrd, Toward more parallel frequent itemset mining algorithms, in Proc. 19th IASTED Int. Conf. Parallel Distrib. Comput. Syst., 2007, pp. 97103.

[3] T. Shintani and M. Kitsuregawa, Mining algorithms for sequential patterns in parallel: Hash based approach, in Proc. 2nd Pacific-Asia Conf., Res. Develop. Knowl. Discovery Data Mining, 1998, pp. 283294.

[4] R. Srikant and R. Agrawal, Mining sequential patterns Generaliza- tions and performance improvements,in Proc. 5th Int. Conf. Extending Database Technol.: Adv. Database Technol., 1996, pp. 117.

[5] R. Kessl and P. Tvrd, Probabilistic load balancing method for parallel mining of all frequent itemsets, in Proc. 18th IASTED Int. Conf. Parallel Distrib. Comput. Syst., 2006, pp. 578586.

[6] V. Guralnik, N. Garg, and G. Karypis, Parallel tree projection algorithm for sequence mining,in Proc. 7th Int. Euro-Par Conf. Euro-Par Parallel Process., 2001, pp. 310320.

[7] S. Cong, J. Han, J. Hoeflinger, and D. Padua, A sampling-based framework for parallel data mining, in Proc.10th ACM SIGPLAN Symp. Principles Practice Parallel Program., 2005, pp. 255265

[8] V. Guralnik and G. Karypis, Dynamic load balancing algorithms for sequence mining, Univ. Minnesota, Minneapolis, MN, US, Tech. Rep. TR 01-020, 2001.

[9] S. Cong, J. Han, J. Hoeflinger, and D. Padua, A sampling-based framework for parallel data mining, in Proc.10th ACM SIGPLAN Symp. Principles Practice Parallel Program., 2005, pp. 255265.

[10] M. J. Zaki, Parallel sequence mining on shared-memory machines, J. Parallel Distrib. Comput., vol. 61, no. 3, pp. 401426, 2001.

[11] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, New algorithms for fast discovery of association rules, in Proc. 3rd Int. Conf. Knowl. Discovery Data Mining, 1997, pp. 283286.

[12] K.-M. Yu and J. Zhou, Parallel TID-based frequent pattern mining algorithm on a PC cluster and grid computing system, Expert Syst. Appl., vol. 37, no. 3, pp. 24862494, 2010.

[13] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, New algorithms for fast discovery of association rules, in Proc. 3rd Int. Conf. Knowl. Discovery Data Mining, 1997, pp. 283286.