INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 6, Issue 2 NOV 2017

Implementing a Parallel Method for Mining of Frequent Sequences Using Static Load-Balancing

[1]MALA SRUTHI, [2]G.UDAY KIRAN

[1]M. Tech Student, Department of CSE, B.V. Raju Institute of Technology, Narsapur, Medak District, Telangana, India

[2]Assistant Professor, Department of CSE, B.V. Raju Institute of Technology, Narsapur, Medak District, Telangana, India

ABSTRACT– ***Revelation of Frequent succession mining is a basic information mining assignment with expansive applications .The yield of the calculation is utilized as a part of numerous different zones like market basket investigation, science and bioinformatics. The incessant succession mining is computationally high costly. Advance consecutive examples mining in a solitary measurement mining and multidimensional successive examples can give us more valuable and helpful examples. Because of the enormous improve in information volume and furthermore genuinely extensive inquiry space, proficient answers for discovering designs in multidimensional succession information are right now imperative. Thus, in this paper, we are building up a frequent pattern mining calculation. Parallel calculation takes after the well ordered approach and every taking part processor or laborers produce competitor arrangement and check their estimate time and supports independently.***

1. INTRODUCTION

The succession mining undertaking is to find an arrangement of properties (things), shared crosswise over time among an expansive number of articles (exchanges) in a given database. The errand of finding every single regular arrangement in expansive databases is very testing. The hunt space is to a great degree substantial. For instance, with m traits there are O(mk) possibly visit groupings of length at generally k. Unmistakably, consecutive calculations can't give adaptability, as far as the information measure and the execution, for expansive databases. We should hence depend on parallel multiprocessor frameworks to fill this part. Past work on parallel succession mining has just taken a gander at circulated memory. In this paper we take a gander at shared memory frameworks, which are fit for conveying elite for low to medium level of parallelism at a monetarily appealing cost. SMP machines are the predominant sorts of parallel machines right now utilized as a part of industry. Singular hubs of parallel appropriated memory machines are additionally progressively being intended to be SMP hubs.

The automated data collection causes that companies own huge databases. The companies are interested in analysing their databases, so they can use them for making better decisions. The process of analysing the data is called data mining. Unfortunately, extreme growth of the database sizes make using ordinary data mining techniques unfeasible. Processing of databases of such sizes is almost impossible with a single processor. Therefore, new parallel algorithms that are able to process such amount of data are needed. Today, large shared-memory machines parallel are still quite expensive. Distributed-memory multiprocessors can be easily built from cheap computers connected with a special network. Therefore, we consider designing algorithms for distributed-memory parallel machines. One of the important data mining tasks is the search for co-occurrences among the data, so called frequent itemset mining. The frequent itemset mining was introduced in the context of analysing the market basket of a consumer in a retail store (hence the term market basket analysis).

Unending case mining is a fundamental data mining methodology with a wide collection of mined cases. The mined unending illustrations can be sets of things (thing sets), progressions, diagrams, trees, et cetera. Customary gathering mining was at first portrayed. The GSP estimation presented and it is the first to handle the issue of general gathering mining. As the ceaseless course of action mining is an expansion of thing set mining, the GSP figuring is a growth of the Apriori computation. The Apriori and the GSP counts are breadth first interest figuring. The GSP computation perseveres with relative issues as the Apriori estimation: it is direct and memory consuming. As a result of the steadiness and memory use of estimations depicted distinctive figuring were proposed. The two significant considerations in the consistent progression mining are those of Zaki and Pei and Han. These two counts use the gathered prefix based indistinguishable quality classes (PBECs in short), i.e., address the case as a string and bundle the course of action of all cases into disjoint sets using prefixes. The two estimations change just in the information structures used to control the request. The calculations depicted are speedy. Regardless, exactly when the continuous computation continues running for a truly lengthy time span there is a necessity for parallel estimations. For instance, the one delineated in this paper, there is an amazingly standard opportunity to parallelize a subjective constant gathering mining estimation: fragment the game plan of each and every customary progression using the PBECs.

2. RELATED WORK

Yaling Xun, Jifu Zhang, and Xiao Qin develop a parallel frequent itemsets mining set of rules known as FiDoop to resolve the scalability and cargo balancing challenges inside the existing parallel mining algorithms for frequent itemsets. FiDoop includes the frequent items ultrametric tree or FIU-tree rather than conventional FP trees, thereby attaining compressed storage and avoiding the necessity to build conditional pattern bases. FiDoop seamlessly integrates three MapReduce jobs to perform parallel mining of common itemsets. The 0.33 MapReduce interest performs an essential role in parallel mining; its mappers independently decompose itemsets even as its reducers assemble small ultrametric trees to be separately mined.

H. D. K. Moonesinghe, Moon-Jung Chung, Pang-Ning Tan e proposed a parallel implementation of the

sequential FP-Tree (and FPGrowth) mining algorithm. The main stages of our algorithm are to build the trees and carry out the mining task in parallel until all the frequent itemsets are generated. They have tested our algorithm using several data sets of size up to 5 millions on a 32-processor distributed memory environment. Their experiments showed good speedups for almost all the cases. Furthermore, it showed scalable performance on the 32-processor environment tested. This was significant for larger data sets, particularly with more than 2 million transactions.

M Ahdi Esmaieli and Mansour Tarafdar theoretically gift a multidimensional sequence version after which use SPMD (Single Program Multiple Data) strategy to parallelize multidimensional sequential sample mining. According to this approach a set of processors execute in parallel the identical algorithm on one-of-a-kind partitions of a dataset. The important goal of the algorithm is balanced workload many of the processors and true scalability. They have simulated their parallel algorithm the use of several data sets on unmarried processor. Simulation outcomes on synthetic facts sets show that if they put into effect this version on actual computing surroundings, we may achieve right speedup.

Sushila S. Shelke, Suhasini A. Itkar studies many current sequential sample mining algorithm which can be applied on allotted environment like Grid, Cloud, Cluster and Hadoop MapReduce. New system is carried out in this paper which uses Hadoop MapReduce surroundings to discover sequential styles. DSPM first converts the series database into transformed database that is represented by directed graph as statistics shape to reduce memory storage and scanning time of database regularly. Because of allotted environment like Hadoop the scaling hassle is stepped forward inside the new device. DSPM is in comparison with UDDAG on single device and it offers higher overall performance the use of directed graph statistics structure.

3. FRAMEWORK

A. Overview of Proposed System

In this paper we proposed is a novel parallel method that statically load-balance the computation. That is: the set of all frequent sequences is first split into Sequential Prefixspan Algorithm (PBECs) and it forms a tree, the relative execution time of each PBEC is estimated and finally the PBECs are assigned to processors. The technique assesses the preparing time of one PBEC by the consecutive Prefixspan calculation utilizing inspecting. It is critical to know that the running time of the consecutive calculation scales with: 1) the database measure; 2) the quantity of frequent sequences; 3) the quantity of embeddings of frequent sequences in database exchanges.

Static load adjusting of the calculation starts with dividing the arrangement of every single frequent sequence into disjoint assignments. Since the PBECs are disjoint, they superbly fit the requirements of the calculation. The aggregate preparing time of the consecutive Prefix span calculation. The preparing time of each PBEC ought to be assessed. The calculation starts part the arrangement of every single frequent grouping into smaller pieces recursively utilizing PBECs. The relative size of a PBEC is the gauge of the division of the aggregate preparing time of a PBEC.

B. Sequential Prefixspan Algorithm

This Sequential Prefixspan Algorithm denoted as PBEC and forms a tree. The operations of this algorithm are;

1. Collecting Frequent Extensions
2. Pseudo-Projected Database Construction

Collecting Frequent Extensions

To collect frequent extensions, we have a sequence, a pseudo-projected transaction as well its corresponding transaction and we need to scan al items at position in the sequence. The complete computational complexity of this collection depends on the database as well as the pseudo-projected database.

Pseudo-Projected Database Construction

We create the initial pseudo-projected transaction for an item for each transaction. We store the positions of item in sequence in the new pseudo-projected database.

Pseudo-Projected Database Construction will possible by other two operations such as;

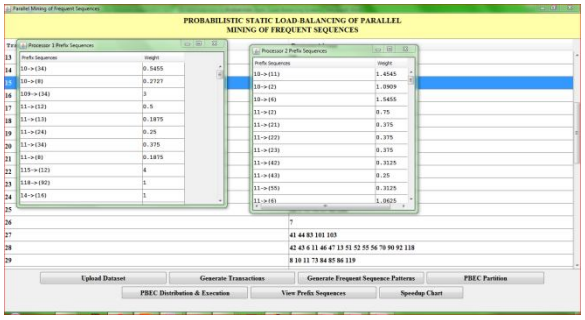
1. Projection using a sequence extension
2. The projection using an event extension

C. Parallel Prefixspan Algorithm

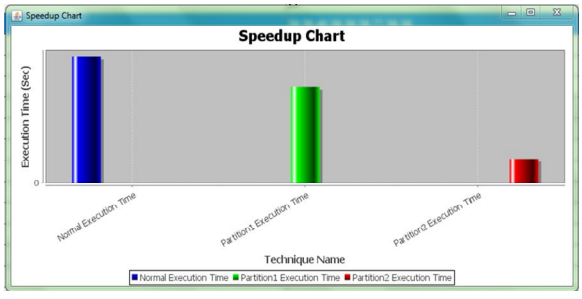
The parallel Prefixspan set of rules has four phases. In the Phase 1, the approach produces the weighting tree containing the estimates of the relative processing time of the PBECs. In the Phase 2, the method walls the set into PBECs, using the tree, and schedule PBECs on processor. In the Phase 3, the approach distributes the database in one of these manner that each processor can process independently its assigned PBECs. In the Phase 4, the method executes the Prefixspan algorithm in parallel on all processors, processing its assigned PBECs.

4. EXPERIMENTAL RESULTS

In this experiment, we used two processors as processor1 and processor2. We have to run these two processors and after we need to upload the LOG dataset. The uploaded dataset contains IPAddress, URLs and time. After, we can generate the transactions and transaction IDs for the uploaded data. And also we can generate the frequent sequence patterns which include an item name, size of transaction, count and support value.



By using PBEC distribution, we can distribute the logs among the two processors. It will display the prefix sequences and weights.



In the Speedup chart, we can observe the execution time of the normal sequence and proposed sequence.

5. CONCLUSION

We proposed an algorithmic program for mining of frequent sequences exploitation static load equalization. The strategy creates a sample of frequent sequences and uses this sample for estimating the relative quantity of the rule inside the PBECs. Assess of the relative amount is in reality performed by estimating method quality of process varied PBECs. The relative interval is then used for partitioning and programming of the PBECs. The matter is that the computable size of a PBEC relies upon at the event of the PBEC. This dependency can be altogether hazard eliminated through exploitation.

REFERENCES

[1] R. Agrawal and J. C. Shafer, "Parallel mining of association rules," IEEE Trans. Knowl. Data Eng., vol. 8, no. 6, pp. 962–969, Dec. 1996.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in Proc. 20th Int. Conf. Very Large Data Bases, 1994, pp. 487–499.

[3] R. Agrawal and R. Srikant, "Mining sequential patterns," in Proc. 11th Int. Conf. Data Eng., 1995, pp. 3–14.

[4] V. Chv atal, "The tail of the hypergeometric distribution," Discr. Math., vol. 25, no. 3, pp. 285–287, 1979.

[5] 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. 255–265.

[6] R. L. Graham, "Bounds on multiprocessing timing anomalies," SIAM J. Appl. Math., vol. 17, no. 2, pp. 416–429, 1969.

[7] D. Gunopulos, R. Khardon, and R. S. Sharma, "Discovering all most specific sentences," ACM Trans. Database Syst., vol. 28, pp. 140–174, 2003.

[8] D. Gunopulos, H. Mannila, and S. Saluja, "Discovering all most specific sentences by randomized algorithms," in Proc. 6th Int. Conf. Database Theory, 1997, pp. 215–229.

[9] 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. 310–320.

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