Parallel Implementation of FP Growth Algorithm on XML Data Using Multiple GPU

Sheetal Rathi and C.A. Dhote

Abstract The FP Growth algorithm is inherently faster than Apriori as it has less number of combinations to be considered. However, the gap here is that the tree building task is a strenuous process in terms of time and memory. Several attempts have been made to improvise the algorithm. In this paper, a model is proposed to implement a parallel FP Growth algorithm that makes use of the elimination process employed by FP Growth algorithm without generating the actual tree (or multiple smaller trees). This not only improves performance of the algorithm but also results in more efficient memory usage. The proposed algorithm Accelerated Frequent Itemset Mining (AFIM) makes use of multiple Graphics Processing Unit (GPU) system.

Keywords Parallel computing · FP growth · Frequent itemset mining · High performance computing · Graphics processing unit

1 Introduction

Data Mining and Knowledge Discovery in Databases (KDD) is a rapidly growing interdisciplinary field. It can also be stated as the process of automated extraction of hidden, previously unknown and potentially useful information from large databases [1]. Association rule mining is an integral task of data mining research domain and it is the process to help find the relationship between the itemset in a large number of databases.

S. Rathi (✉)
SGBAU, Amravati, Maharashtra, India
e-mail: sheetal.rathi@thakureducation.org

C.A. Dhote
PRMITR, Badnera, Maharashtra, India
e-mail: vikas.dhote@rediffmail.com

© Springer India 2015

J.K. Mandal et al. (eds.), *Information Systems Design and Intelligent Applications*,
Advances in Intelligent Systems and Computing 339,
DOI 10.1007/978-81-322-2250-7_58

Apriori [2], Eclat [3] and FP Growth [4] are the major vital algorithms used for association rule mining. Research suggest that the methodologies mainly used rely heavily on an Apriori-like [5] candidate set generation-and test approach. However, the approach fails when the dataset is very large. Another important methodology is EClat [3] that proved a substantial milestone in the development of association rule mining process. The third approach towards association rule mining is using FP Growth method proposed by Han and Pei [4]. The frequent pattern (FP) growth uses only 2 database scans irrespective of the length of longest transaction, whereas the Apriori needs n + 1 scans (n: length of the longest pattern).This significantly reduces computation time and cost especially when the number of candidate itemsets are more. However, because of its flat tree like structure, the basic FP growth algorithm fails miserably when it comes to storage space. To speed up the process, many researchers have directed their research in developing parallelized variants of the method [6–10].

In this paper, we propose a highly parallel FIM algorithm that is optimized for a platform consisting of GPUs and CPU. The key idea is to use the trimming feature of the FP-Growth Algorithm used after the tree generation before even the tree is generated. All the transactions are kept separately and not combined into a tree.

2 Related Work

Literature suggests that to mine association rules efficiently, the frequent itemset mining step has to be accelerated effectively because this is the most time consuming step. Enormous amount of work has been done to speed up this process.

2.1 Parallel Approach

High-performance parallel and distributed computing environment is becoming increasingly important, as data continues to grow inexorably in size and complexity. Several efforts have been applied to parallelize the mining process such as shared-memory systems (SMPs), distributed-memory systems, hybrid systems consisting of a cluster of SMPs and geographically distributed systems [11, 12].

In [10], the authors designed a parallel algorithm to work in distributed data framework. It does not need to create the overall FP tree because of which it can handle scalable data. A modern multi core processor is used to implement parallel FP tree algorithm [13]. Two techniques were used: a cache-conscious FP-array and a lock-free dataset tiling parallelization mechanism. Zang et al. [14] presented two parallel FIM algorithms, Apriori and Eclat on shared memory platform. The strategy was to implement three different vertical data representations; vertical transaction id set, vertical bitvector, and diffset. They concluded that Apriori is only scalable when used with diffset while Eclat is generally scalable but achieves its best scalability with diffset.

2.2 GPU Acceleration

Bakos et al. [15] developed a new parallel Frequent Itemset Mining algorithm called "Frontier Expansion" developed from Eclat. High performance is achieved by using a heterogeneous platform consisting of a shared memory multiprocessor and GPU coprocessors. Here a partial breadth-first search is utilized to make use of extreme parallelism while being controlled by the accessible memory capacity. Authors concentrate on the task of scaling a MapReduce application using the CPU and GPU together in a combined architecture [16]. Different methods like map-dividing scheme and pipelining were developed for dividing the work. Dynamic work distribution schemes were devised for both the approaches. Their implementation of MapReduce is based on an endless reduction method, which avoids the memory overheads of storing key-value pairs. Huang et al. [9] utilizes the computing power offered by GPUs and extend CPU-based data mining algorithms for mapping to CPU-GPU hybrid and scalable architecture for speeding up the computing process.

3 Preliminaries

3.1 Problem Definition

The association rule mining is a two-step process:

1. Find all frequent itemset having minimum support.
2. Generate strong rules having minimum confidence, from the frequent itemset.

3.2 Basic FP Growth Algorithm

Frequent patterns are mined from an FP-tree using a method called FP-growth. All the node-links from the header table are traversed to generate complete frequent patterns. The general idea is to create conditional pattern bases depending on the support threshold and in turn generate rules based on minimum confidence.

```
1) IF Tree contains a single path P THEN
2) FOR all each β= nodes combination in P, DO
3) generate pattern=βUα with
    Support= minsup of nodes in β;  ELSE
4) FOR all αi in the header of Tree  DO
BEGIN
5) generate pattern β= αiUα with support=αi.support;
6) Construct conditional pattern base of β
```

```
Treeβ= construct conditional FP-tree of β;
7) IF Treeβ≠ϕ THEN
8) CALL FP-growth (Treeβ, β);
9) END;
```

3.3 Graphics Processing Unit

In this paper we exploit the highly parallel computation power of GPU for frequent itemset mining process. Unlike multi-core CPUs, the cores on the GPU are virtualized, and GPU threads are executed in Single Instruction, Multiple Data (SIMD) which are in turn managed by the hardware. Because of the embedded hardware based design, the GPU programming is cut down and also improves program scalability and portability. The main reason behind this is that programs are unaware about physical cores and rely on hardware for thread management (Fig. 1).

The inspiration behind using General purpose graphical processing unit (GPGPU) is its cost effectiveness as compared to the distributed and parallel approaches implemented so far [10, 14, 17].

4 Framework Architecture of AFIM

Scalable GPU-based parallel model for speeding up the frequent itemset mining (FIM) process is implemented in this paper. The purpose of doing this is to best exploit the computing resources obtained by GPUs and extend traditional, CPU-based data mining algorithms for mapping to CPU-GPU hybrid architecture,.

The idea to be implemented in this algorithm is to apply Accelerated Frequent Itemset Mining (AFIM) algorithm and use the trimming feature of the FP-Growth

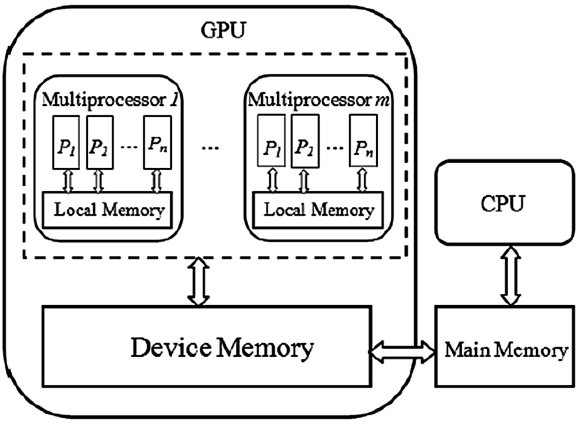

Fig. 1 The many core architecture of GPU (*Source* [18])

Algorithm used after the tree generation before even the tree is generated. All the transactions are kept separately and not combined into a tree. The entire process can be summarized as follows:

1. The arrays of individual transactions are grouped into different sections based on the initial item in each transaction i.e. all transactions with the same item at the start are grouped together in one section. Thus, multiple such sections are formed based on the occurrence of the most frequently occurring items in most transactions. Once grouped, that item is considered the first part of the pattern. The next part is thus identified by repeating the same process over the new section formed. The items of the subsection are then appended to the original first item of the given section.
   Here, each section of the transaction basically represents 1 level of the tree. The 0th partition i.e. no partition represents all the transactions. The 1st partition represents the items formed at level 1 of the FP tree i.e. it identifies those transactions which have most frequent item in the first place of each transaction. It is to be noted that each partition may result in multiple sections based on the variance in the transaction set.
2. Elimination takes place in this step by discarding those item patterns/transactions which do not have sufficient count to meet the threshold criteria.
3. Each section then undergoes the same process of partitioning and identifying the different items that occur after the dominant item of that given section.
4. At this point, the CPU forces the GPU for synchronizing all threads.
5. The above process is iteratively repeated for every subsection as GPUs are not suited for recursion. This ends up locating the most frequent and obvious patterns in the transactions. In the GPU implementation, we assign different section to different blocks in a GPU. Each block then individually assesses its own section and identifies recurring patterns in its section.

Each section when formed has two roles to play;

1. The dominant item i.e. the item in the 1st position of all the transactions in that given section is identified. The remaining part of each transaction in the section is then subjected to further partitioning and
2. On receiving the frequent pattern found by the subsections of a section, the section merges it with the frequent item that it has identified and passes it on up to the section of which it is a subsection.

As can be seen, the entire process appears very tree like but the actual FP tree is not generated. The process flow diagram is as shown in Fig. 2.

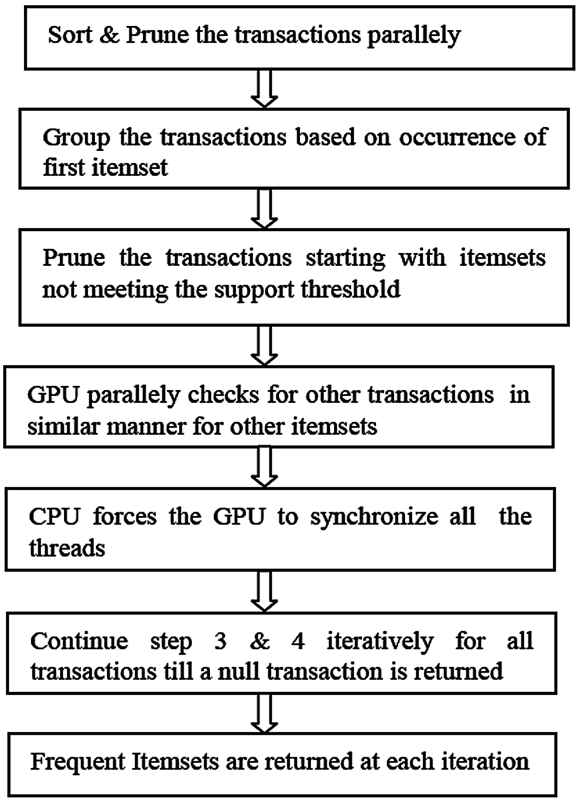

Fig. 2 Proposed AFIM model

5 Experimental Result and Analysis

Various synthetic data sets were generated using Python script for testing the AFIM algorithm. Again a fully optimized CPU is used for serial computation and a GPU-CPU model is used for the parallel part.

The system configuration used is Intel Xeon E5-2620 (6 cores, 12 threads, and clock speed of 2 GHz), RAM: 15 GB GPU: Tesla K40c (GDDR5 SDRAM). The characteristics of the dataset were varied in such a way that for first three datasets, i.e. T20I100D50K, T20I100D1M, and T20I100D5M only the dataset size is increased and the performance is evaluated. In the next three datasets, T25I120D50K, T25I120D1M and T25I120D5M, the itemset count and transaction length is differed. The driver used was the NVIDIA driver for Ubuntu 10.04 (CUDA compilation tools, release 5.0, VO.2.1221) with CUDA support driver version NVIDIA-SMI 331.67. Table 1 elaborates the speed up of CPU-GPU implementation with respect to the CPU based implementation.

It is evident that there is considerable speed up for AFIM model as compared to core CPU implementation which is clearly depicted in Fig. 3. However, even if the

Table 1 Performance evaluation of CPU and GPU-CPU

Synthetic dataset	Processing time (CPU)	Processing time (GPU-CPU)	Speed up
T20I100D50K	2,425	769	3.15
T20I100D1M	11,453	1,635	7.00
T20I100D5M	13,452	2,089	6.43
T25I120D50K	3,524	854	4.12
T25I120D1M	12,539	1,687	7.43
T25I120D5M	13,869	2,135	6.49

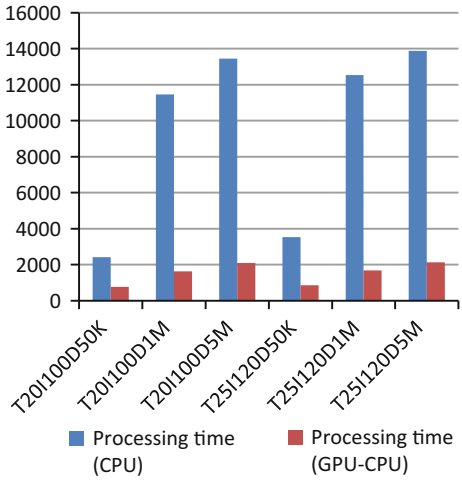

Fig. 3 Performance evaluation of CPU and CPU-GPU architecture

characteristics of datasets are changed, there is no considerable difference in the amount of processing time take, both by CPU and GPU for the same size of dataset.

In above figure, we have compared the AFIM algorithm with the fully optimized CPU on the same machine configuration. Synthetic datasets with varying characteristics were tested on the two processors, CPU and CPU-GPU architecture.

6 Conclusion

Association Rule mining has been parallelized for a long time with expensive system setup. This paper has aimed to perform Association Rule mining using parallel processing with the help of Graphic processors, which have in-built hardware capability of multi-threading and that has been leveraged for our algorithm.

Previous work does not prevail the use of GPU for FP-growth algorithm. Instead literature speaks about the various approaches exploited to achieve parallelism in FP-growth using shared and distributed processors. In this paper we have

parallelized FP-growth approach on multi-threaded Graphic Processor. There was considerable speed up on different synthetically generated datasets with different characteristics. In addition to the speed boost, the dataset size also progressed. Initially starting from 100 transactions it succeeded in achieving scale up in data size of up to 5 million transactions.

Acknowledgments This project is based upon work supported by the CUDA Centre of Excellence (CCOE), IIT Bombay.

References

1. Kamber, M., Han, J.: Data Mining: Concepts and Techniques, 2nd edn. Morgan Kaufmann Publishers, Burlington (2006)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1994)
3. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Hsu, M.-C.: Mining sequential patterns by pattern-growth: the prefix span approach. IEEE Trans. Knowl. Data Eng. **16**(10), 1–17 (2004)
4. Han, J., Pei, J.: Mining frequent patterns by pattern-growth: methodology and implications. SIGKDD **2**(2), 30–36 (2000)
5. Zaiane, O.R., El-Hajj, M., Lu, P.: Fast parallel association rule mining without candidacy generation. In: Proceedings of IEEE International Conference on Data Mining ICDM, pp. 665–668 (2001)
6. Hu, J., Yang-Li, X.: A fast parallel association rules mining algorithm based on FP-forest. In: Part II LNCS, pp. 40–49. Springer, Berlin (2008)
7. Chen, M., Gao, X.D., Li, H.F.: An efficient parallel FP-growth algorithm. In: Proceedings of International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pp. 283–286 (2009)
8. Orlando, S., Palmerini, P., Perego, R., Silvestri, F.: 5th International Conference Porto, Portugal (2002)
9. Huang, Y.-S., Yu, K.-M., Zhou, L.-W., Hsu, C.-H., Liu, S.-H.: Accelerating parallel frequent itemset mining on graphics processors with sorting LNCS vol. 8147, pp. 245–256. IFIP International Federation for Information, Springer, Berlin (2013)
10. Wang, Z., Wang, C.: A parallel association-rule mining algorithm. In: WISM'12 Proceedings of the 2012 International Conference on Web Information Systems and Mining, pp. 125–129. Springer, Berlin (2012)
11. Vikram singh, T., Robil, V.: Data mining by parallelization of FP-growth algorithm. Int. J. Eng. Res. Dev. **5**(9), 0–35 (ISSN: 2278-067X) (2013)
12. Zaki, M.J.: Parallel and distributed association mining: a survey. IEEE Concurrency **7**(4), 14–25 (1999)
13. Liu, L., Li, E., Zhang, Y., Tang, Z.: Optimization of frequent itemset mining on multiple-core processor. In: Proceedings of VLDB 07.acm, pp. 1275–1285, 23–28 Sept 2007
14. Zhang, Y., Zhang, F., Bakos, J.: Frequent itemset mining on large-scale shared memory machines. In: Proceedings of IEEE International Conference on Cluster Computing, pp. 585–589 (2011)
15. Bakos, J.D., Zhang, F., Zhang, Y.: Accelerating frequent itemset mining on graphics processing unit. J. Supercomput. **66**(1), 94–117 (2013). doi:10.1007/s11227-013-0887-x

16. Agrawal, G., Chen, L., Huo, X.: Accelerating MapReduce on a coupled CPU-GPU architecture. IEEE SC12. Salt Lake City, Utah, USA (2012). 978-1-4673-0806-9/12
17. Böhm, C., Noll, R., Plant, C., Wackersreuther, B., Zherdin, A.: Data mining using graphics processing units. In: Transactions on Large-Scale Data-and Knowledge-Centered Systems I LNCS, vol. 5740, pp. 63–90. Springer, Berlin (2009)
18. Fang, W., Lau, K.K., Lu, M., Xiao, X., Lam, C.K., Yang, P.Y.: Parallel data mining on graphics processors. Technical Report HKUSTCS0807 (Oct 2008)