Accelerating frequent itemsets mining on data stream: a proposal

Lázaro Bustio-Martínez[1,2], René Cumplido-Parra[2], José Hernández-Palancar[1], and Claudia Feregrino-Uribe[2]

[1] Advanced Technologies Application Center.
7^a ♯ 21812 e/ 218 y 222, Rpto. Siboney, Playa, C.P. 12200, Havana, Cuba.
{lbustio,jpalancar}@cenatav.co.cu
[2] National Institute for Astrophysics, Optics and Electronic.
Luis Enrique Erro No 1, Sta. Ma. Tonantzintla, 72840, Puebla, México.
{rcumplido,cferegrino}@ccc.inaoep.mx

Abstract. Frequent itemsets mining is a widely used technique in Data Mining and has application in many areas of common life: data streams analysis is one of such applications. Several sequential and parallel algorithms have been proposed but traditional approaches are ineffective to face data streams, due to the speed of data in such data source and the impossibility of access them twice or more. In this paper we propose a research driven to obtain parallel algorithms capable to perform frequent itemsets mining on data streams using hardware reconfigurable.

1 Introduction

In Data Mining is very useful to record all the occurrences of certain patterns to be used in forthcoming tasks. One of such tasks is frequent itemsets mining where frequent itemsets are those sets of data items that can be found always together (without concerning the apparition order) more than a given number of occurrences in some data source. In other words, the goal of frequent itemsets mining is to determine which itemsets in a database (or any other data source) commonly appear together.

In early 90's Agrawal proposed an algorithm for frequent itemsets mining named *Apriori*[1]. Apriori was the first approach proposed for this task and also the simplest. The process to find all frequent itemsets are computationally intense. Plenty of algorithms have been created to perform frequent itemsets mining tasks. All of these algorithms requires a lot of powerful computational resources and time to solve the combinatorial explosion of itemsets that can be found in a dataset. When it is processing very large datasets, this issue get worst provoking the task can not be achieved and consequently they fail. This is mainly due to the presence of thousands of different patterns or the use of a too low threshold of support (see *Support* concept in 3).

In addition, in recent times there has been a data explosion and classical approaches (like frequent itemsets mining) to extract hidden knowledge from those huge amounts of data fail while trying to process them. Data generating

Special Issue: Advances in Pattern Recognition
Research in Computing Science 61, 2013, pp. 3-12
Paper Received 20-03-2013 and Accepted 22-04-2013

rate is growing exponentially while data mining applications performance has only increased by 10-15% [10]. This fact shows that the classical paradigm of these algorithms is not suitable.

One scenario that is gaining a lot of attention of researchers is the data streams analysis. A data stream can be seen as unordered and potentially infinite flow of data at high velocity rates. Analizing data streams is an emerging need and it can be found in video and audio streams, network traffic, commercial transactions, etc, but those applications need to be as fast as they can so hardware-based approaches have been proposed. Frequent itemsets mining in such scenario addresses new challenges and only in [4] is conducted a research to frequent itemsets mining on data streams.

This paper is structured as follow: in the next section, different platforms to implement algorithms are explained. Section 3 explains the theoretical basis of the frequent itemset mining. A review of state-of-the-art is addressed in section 4 while section 5 presents the proposed research objectives. This paper is concluded in section 6.

2 Platforms for algorithms implementation

There are two main approaches to implement algorithms in general. The first one consists in building Application Specific Integrated Circuits (ASICs). They are designed and built specifically to perform a given task, and thus they are very fast and efficient. ASICs can not be modified after fabrication process and this is their main disadvantage. If an improvement is needed, the circuit must be re-designed and re-built, with the costs that this entails. The second one consists in using a General Purpose Processor (GPP) which is programmed by software; it executes the set of instructions that are needed by algorithms. Changing the software instructions implies a change in the algorithms behavior. This results in a high flexibility but the performance will be degraded. To accomplish certain function, the GPP, first must read from memory the instructions to be executed and then decode their meaning into native GPP instructions to determine which actions must be done. Translating the original instructions of an algorithm introduces a delay.

FPGAs can be seen as a mesh of basic logic gates interconnected together and its functionality is customizable at runtime. The connections between the logic gates are also configurable. The architecture of a FPGAs is based on a large number of logic blocks which perform basic logic functions. Because of this, an FPGA can implement from a simple logical gate, to a complex mathematical function. FPGAs can be reprogrammed, that is, the circuit can be "erased" and then, a new architecture that implements a brand new algorithm can be implemented. This capability of the FPGAs allow the creation of fully customized architectures, reducing cost and technological risks that are present in traditional circuits design. Fig. 1 represent the position of FPGAs between ASICs and GPP.

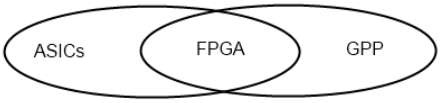

Fig. 1. FPGAs combines the advantages of ASICs and GPP.

3 Theoretical basis

Frequent itemsets mining was introduced by Agrawal et al. back in 1993 [1] and it is used for finding common and potentially interesting patterns in databases. In this scope the data are represented by means of transactions, each of which is a set of items labeled by a unique ID. The purpose of frequent itemsets mining is to find the most frequently-occurring subsets from the transactions. The frequency of the subset is measured by support ratio, which is the number of transactions containing the subset divided by the total number of transactions in the database. Formalizing, let I be a set of items:

Definition 1 (Itemset). *A itemset X is set of items over I such $X = i_i, ..., i_k \subseteq I$. If a set X contains k items then the set X is called* k-itemset. *Normally is considered that the items in an itemset are lexicographically ordered.*

Definition 2 (Transaction). *A transaction over I is a couple $T = (tid, I)$ where tid is the transaction identifier and I is an $X \subseteq I$ itemset. A transaction $T = (tid, I)$ is said to support an itemset $X \subseteq I$, if $X \subseteq I$.*

Definition 3 (Support). *The support of an itemset X in D is the number of transactions in D that supports X:*

$$Support(X, D) = |\{tid | (tid, I) \in D, X \subseteq I\}| \tag{1}$$

An itemset is called $frequent$ if its support is no less than a given absolute minimal support threshold ϕ_{abs}, with $0 \leq \phi_{abs} \leq |D|$. The frequent itemsets discovered does not reflect the impact of any other factor except frequency of the presence or absence of an item.

3.1 Issues in frequent itemsets mining.

The theoretical basis of frequent itemsets mining that has been presented can be applied to both databases and data streams scenarios. The major problem with frequent itemsets mining methods in both scenarios is the explosion of the number of results, so it is difficult to find the most interesting frequent itemsets. Normally must be faced the following disadvantages (databases and data streams scenarios): (a) huge data volumes (order to gigabytes); (b) elevate number of transactions; (c) limited computing resources (like memory) and (d) unpractical processing times.

The main issue in databases (and also in data streams) scenario is concerned to the high number of items to handle (and memory consumption). Let be n the number of single items in database, the number of candidates frequent itemsets is 2^n, or the same, this problem has computational complexity of $O(2^n)$. Handling those huge amount of data is a challenging task, and strategies for efficient data access and data memory maintaining are needed [12].

In other kind of applications, data streams are becoming too attractive and they have been used in many real life applications. Data streams can be defined as a continuous, ordered and potentially infinite sequence of items in real time and also share the same limitations of frequent data streams on databases. Considering the three characteristics of data streams (continuity, expiration and infinity), new limitations are added to frequent itemsets mining problem on data streams: (a) It is impossible to store the stream for latter processing; (b) items in a data stream must be processed just once and (c) items must be processed in a very short time interval.

In other words, frequent itemsets mining on data streams is a particular case of frequent itemsets mining on databases that includes some extra challenges, besides the issues that entails frequent itemsets mining on databases.

4 Algorithms review in hardware

Hardware implementations of algorithms take advantage of inner parallelism of hardware device. In consequence such devices gain every day more attention to be used as development platforms. After a proper review of the state-of-the-art, it can be organized as shown table 1.

Analyzing the revised literature we can realize that frequent itemsets mining on data streams using hardware reconfigurable is an interesting research area that has not been studied in deep. So, we consider that it is worth to propose new parallels algorithms to face such task.

4.1 Frequent itemsets mining based on hardware acceleration

Algorithms for frequent itemsets mining can be divided into two groups: Apriori-based and FP-Growth-based.

Apriori-based. The algorithms that follow the Apriori-based scheme in hardware require loading the candidates itemsets and the database into the hardware. This scheme is limited by the capacity of the chosen hardware platform: if the number of items to handle is larger than the hardware capacity the items must be loaded separately in many consecutive times degrading performance. In consequence, the support counting must be executed several times. Since the time complexity of those steps that need to load candidates itemsets or database items into the hardware is in proportion to the number of candidates itemsets and the number of items in the database, this procedure is very time consuming.

Table 1. Principal algorithms and architectures for the frequent itemsets mining problem on data streams.

Title	**Author**	**Year**	**Based**	**Source**
An Architecture for Efficient Hardware Data Mining Using Reconfigurable Computing Systems. [2]	Baker	2006	Apriori	BD
Hardware Enhanced Mining for Association Rules. [5]	Liu	2006	Apriori	Stream
Hardware-Enhanced Association Rules Mining With Hashing and Pipelining. [11]	Wen	2008	Apriori	BD
Novel Strategies for Hardware Acceleration of Frequent Itemset Mining With the Apriori Algorithm.[10]	Thöni	2009	Apriori	BD
Mining Association Rules with Systolic Trees.[8]	Sun	2008	FP-Growth	BD
A Reconfigurable Platform for Frequent Pattern Mining.[7]	Sun	2008	FP-Growth	BD
A Highly Parallel Algorithm for Frequent Itemset Mining. [6]	Mesa	2010	FP-Growth	BD
Design and Analysis of a Reconfigurable Platform for Frequent Pattern Mining. [9]	Sun	2011	FP-Growth	BD

In addition, numerous candidates itemsets and a huge database may cause a bottleneck in the system.

In the revised literature was found only one paper referred to frequent itemsets mining on data streams scenario that uses hardware reconfigurable. Liu et al. in [5] proposed a hardware-enhanced mining framework and an Apriori-like algorithm to mine frequent temporal patterns[3] from data streams. This architecture is specially designed to mine those itemsets of length 1 and 2, because the computing of L1- and L2-itemsets is the most time-consuming task in their algorithm, so they proposes offload this operation to hardware to enhance the global performance. They states in their work the main issues when dealing with data streams, which are consistent with that was explained in section 3.1. According to the authors, the novelty in this hardware-enhanced approach resides in the transformation of the items transactions from a data streams into a matrix structure and efficiently map operations for discovering frequent items to highly efficient hardware processing unit. A "receiving-storing-processing" approach is used to perform the transformation of transactions of stream into matrix. Experiments on synthetic data set shown that the throughput is two order of magnitudes larger than its software counterpart does.

[3] Frequent Temporal Pattern is referred to those items or itemset that are frequent in some time period.

FP-Growth-based. FP-Growth is one of the fastest and efficient algorithm in the frequent itemsets mining state-of-the-art, which has been implemented in several ways, including sequential and parallel implementations. FP-Growth is based on a prefix tree representation of the given database of transactions (called an FP-tree). The FP-tree representation allow saving considerable amount of memory for storing the transactions. In FP-tree representation every transaction are stored as a string in the tree along with its frequency. Fig. 3 shows the FP-tree presentation for transactions shown in Fig. 2.

ID	Items
1	B, C, D
2	B, C
3	A, C, D
4	A, C, D
5	A, B, C
6	A, B, C
7	A, B, D

Fig. 2. Transaction BD

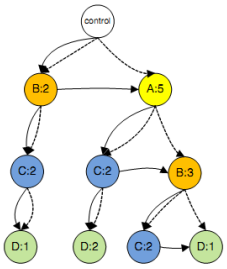

Fig. 3. FP-tree representation of BD in Fig. 2

In 2008 Song Sun and Joseph Zambreno proposed an architecture [8] to speed up the association rules mining process based on FP-Growth. To emulate the FP-Tree data structure they proposed a new hardware structure named *systolic tree*[4]. A systolic tree could be seen as a tree where each node is a processing unit which has their own logic. Fig. 4 shows the systolic tree built for the FP-tree represented in Fig. 2.

The main idea implemented in this architecture is to build a lexicographic tree while items flows through the systolic tree. When all the database is mapped into the systolic tree, each PE will contain the frequency of the current item.

Also in 2008, Sun and Zambreno propose a new hardware architecture for frequent itemsets mining using a systolic tree [7]. As similar with [8] the goal of this architecture is to emulate the original FP-Growth algorithm while achieving a much higher throughput. The main apport in this paper is that Sung et al. modifies the original scheme introduced in [8] by eliminating the counting nodes (see Fig. 2), and providing a new count mode algorithm.

In 2010 Mesa et al. [6] proposed a novel architecture that used FP-Growth as starting point. They proposed a vertical bit vector data layout to represent items and transactions. This layout allows to calculate the support by using logical AND and OR operations. Mesa et al. define a two-dimensional matrix to

[4] In VLSI terminology, a systolic tree is an arrangement of pipelined processing elements in a multi-dimensional tree pattern.

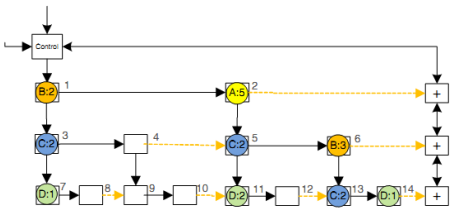

Fig. 4. Systolic tree architecture that emulates the FP-tree data structure represented in Fig. 2.

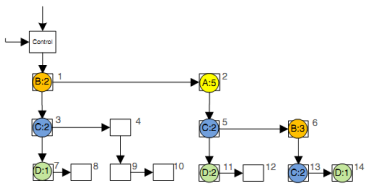

Fig. 5. Enhanced systolic tree architecture that emulates the FP-tree data structure represented in Fig. 2. Notice that the Counting PE was eliminated in this approach.

store the database where the columns represent the elements of the dataset and the rows represent the transactions. Using such data representation, Mesa et al. proposed an algorithm that is based on a search over the solution space through the equivalence class considering a lexicographical order over the items. This is a two-dimensional search (bottom-down, right-left), both breath and depth is performed concurrently. The algorithm proposed do not need a candidate generation stage and it uses a binary tree structure of processing elements. This structure represent a systolic tree and it was chosen because it allows to exponentially increase the concurrent operations at each processing step. For the chosen device, only 11 items can be processed. Experiments demonstrates that the proposed architecture outperforms the architecture reported in [7] almost in one order of magnitude. Also, the architecture performs better when the density of the database and the number of frequent itemsets increases.

Once again, in 2011 Sun et al. went back to [7] and explain in a more detailed manner the systolic tree architecture and their approach [9]. In this paper the authors expose the same idea reported in [7], but they extend their paper with more detailed explanations about systolic tree and the working modes of their architecture. Also, new experiments were performed demonstrating that the systolic tree architecture can outperform the best known FP-Growth implementation [3]. Experiments demonstrate that systolic tree is a valid architecture to mine frequent patterns. For those dataset having sizes that can be placed directly in the device, the systolic tree architecture always outperform FP-Growth. In such cases that the dataset could not be placed directly in the FPGA, a dataset projection must be used and chosen. In this work, Sun et al. proposes one projection dataset strategy and prove its feasibility. In such cases, FP-Growth outperform systolic tree structure. This behavior is caused by the overhead introduced by the projection strategy for database transaction mapping over FPGA. If the overhead is not amortized by the runtime reduction, the systolic tree algorithm is slower than the original FP-Growth algorithm.

In this research proposal, we will focus on FP-Growth-oriented and data-stream-oriented algorithms because of this approach is closer to hardware nature and in the mining process, items are accessed only once.

4.2 Research Problem

Modern applications generate huge data volumes in a data streams way. Due to the increase of this kind of applications it is necessary obtain useful knowledge from those data streams. As it was previously defined, a data streams are a continuous, ordered and potentially infinite sequence of items in real time where data arrives without interruptions at a high speed. Also, data can be accessed only once and the only assumption that we can make about bounds of streams is that the total number of data is unbounded. It is unrealistic to store all items of data streams to process them offline. These characteristics impose an extra difficulties to algorithms and systems that process such data sources.

Due to the high incoming rate, the impossibility to store the data and the huge volumes of items in streams, softwares that analyze they can not process exhaustively all items. The supporting hardware and software are not capable to deal with such intense processing. Instead, softwares use an "approximate" processing approach. That is, they do not analyzes all and each one of items that are present in a flow; instead they use some heuristic or probabilistic approach to determine which item is the most likely to contain the desired information. There are some applications where this approach is not valid, for example in an intrusion detection system or network analysis system. In these kind of applications, the exhaustive data analysis and realtime response are extremely valuable. To provide an realtime response and an exhaustive item processing is needed to use a more specific hardware such as FPGAs. FPGAs can perform tasks in a high parallel fashion and this allow to process all items in a data streams exhaustively in realtime.

Frequent itemsets mining is one technique that is very used in data knowledge extraction and have been used with success in data streams scenario. The frequent itemsets mining on data streams using software is performed in an approximate way. To mine frequent itemsets on data streams exhaustively and responding in realtime, is needed to develop new parallel algorithms and the use of custom hardware architectures. In the reviewed literature there is only one architecture to mine frequent itemsets on data streams [4].

Summarizing, data streams are a modern data source that are gaining interest in recent applications. Traditional approaches are not suitable to be used in data streams and they introduce new challenges to frequent itemsets mining. Due to the growing number of applications that produces data streams and the impossibility to process them in a proper manner, it is worth to propose algorithms that can analyze data streams in real time. To pursuit such goal, we propose to use FPGAs as development platform taking advantage of the parallel capabilities of FPGAs.

4.3 Aims

The general aim of this research work is :

To develop new methods for frequent itemsets mining on data streams that outperform the state-of-art algorithms in data streams analysis in a more efficient and effective manner.

The specific aims proposed are:

1. To select an optimal method to perform the transactions separation in a data streams.
2. To obtain a new method for items representation that can be used in frequent itemsets mining on data streams.
3. To develop new algorithms for frequent itemsets mining that use the separation method and items representation proposed earlier that can outperform the state-of-art algorithms in a more efficient and effective manner.
4. To obtain a parallel hardware implementation of the specific objectives mentioned above that can perform frequent itemsets mining at least 10 times faster (without compromising efficiency) than state-of-the-art software implantations, and 2 times faster than state-of-the-art hardware implementations.

4.4 Contributions

The contributions expected of this proposal are:

1. A new compact data representation that can be used in data stream handling.
2. A design of a parallel one-pass algorithm to mine frequent itemsets on data streams that uses the compact data representation presented before.
3. A parallel implementation in hardware (specifically in FPGAs) of the proposed algorithm.
4. An analysis of the performance of the proposed algorithm and an analysis of the tradeoff between parallelism-performance that allows to estimate the scalability of the implemented system.

5 Conclusions

Frequent itemsets mining is a widely used technique in Data Mining applications. Hence, many researchers are currently engaged in proposing methods for improve and accelerate such process while the data sources increase their sizes. In recent times the streams of data have gained more interest due to its applications, and frequent itemsets mining are being used to obtain new knowledge from those unorganized and continuous data flows.

Classical methods are ineffective when they deal with data streams so it is necessary to explore other alternatives methods such us parallel algorithms that take fully advantages over hardware devices like FPGAs. Data flows can be seen

as an especial case of data bases except that data flows introduce some restrictions. So, the aim of this work is to develop new parallel algorithms than can perform the frequent itemsets mining over data streams using reconfigurable hardware that outperform reported reported results for both software and hardware approaches.

References

1. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499. Morgan Kaufmann Publishers Inc., 1994.
2. Zachary K. Baker and Viktor K. Prasanna. An architecture for efficient hardware data mining using reconfigurable computing systems. In *Proceedings of the 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, FCCM '06. IEEE Computer Society.
3. Christian Borgelt. An implementation of the fp-growth algorithm. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, OSDM '05, pages 1–5. ACM, 2005.
4. Chih hsiang Lin, Ding ying Chiu, and Yi hung Wu. Mining frequent itemsets from data streams with a time-sensitive sliding window. In *In SDM*, 2005.
5. Wei-Chuan Liu, Ken-Hao Liu, and Ming-Syan Chen. Hardware enhanced mining for association rules. In *Proceedings of the 10th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining*, PAKDD'06, pages 729–738. Springer-Verlag, 2006.
6. Alejandro Mesa, Claudia Feregrino-Uribe, Ren Cumplido, and Jos Hernndez-Palancar. A highly parallel algorithm for frequent itemset mining. In J.F. Martnez-Trinidad, J.A. Carrasco-Ochoa, and J. Kittler, editors, *Advances in Pattern Recognition*, volume 6256 of *Lecture Notes in Computer Science*, pages 291–300. Springer Berlin Heidelberg, 2010.
7. Song Sun, Michael Steffen, and Joseph Zambreno. A reconfigurable platform for frequent pattern mining. In *Proceedings of the 2008 International Conference on Reconfigurable Computing and FPGAs*, RECONFIG '08, pages 55–60. IEEE Computer Society, 2008.
8. Song Sun and Joseph Zambreno. Mining association rules with systolic trees. In *FPL*, pages 143–148. IEEE, 2008.
9. Song Sun and Joseph Zambreno. Design and analysis of a reconfigurable platform for frequent pattern mining. *IEEE Transactions on Parallel and Distributed Systems*, 22:1497–1505, 2011.
10. David W. Thöni and Alfred Strey. Novel strategies for hardware acceleration of frequent itemset mining with the apriori algorithm. In *19th International Conference on Field Programmable Logic and Applications, FPL 2009*, pages 489–492. IEEE, 2009.
11. Ying-Hsiang Wen, Jen-Wei Huang, and Ming-Syan Chen. Hardware-enhanced association rule mining with hashing and pipelining. *IEEE Trans. on Knowl. and Data Eng.*, 20(6):784–795, June 2008.
12. Guizhen Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 344–353. ACM, 2004.