Hardware Acceleration of Frequent Itemsets Mining on Data Streams

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

[1] Advanced Technologies Application Center.
7^a ♯ 21812 e/ 218 and 222, Rpto. Siboney, Playa, C.P. 12200, Havana, Cuba.
{lbustio,rhernandez}@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. In recent times, processing of data streams is gaining the attention of the scientific community due to its practical applications. Data stream is an unbounded and infinite flow of data arriving at high rates and, therefore, the classical data mining approaches can not be used straightforward in this scenario. Because of this, finding alternatives to achieve better results in the discovering of frequent itemsets on data streams is an active research topic. One of such alternatives is to develop single-pass parallels methods that can be implemented in hardware to take advantage of the inner parallelism of such devices. In this paper, a new method that can mine high incoming rates data streams is presented. As preliminary results, the proposed methods can mine in exhaustive fashion the incoming data streams when its number of single items is low. When the number of single items is high, the proposed method obtains an approximate solution with no false positives itemset produced.

Key words: Frequent itemset mining, data stream mining, systolic tree, custom hardware architectures.

1 Introduction

In recent years, there has been an explosion on the amount of data generated by all sort of human activities. In order for this data to be useful, it must be processed to obtain hidden knowledge. To perform this task, several approaches have been proposed and implemented mainly in software-based systems that offer limited performance when processing large amounts of data.

Data Mining aims to pride the tools and techniques needed to face such immense data volumes. In Data Mining is extremely useful to record all the occurrences of certain patterns and that is what frequent itemsets mining performs. Frequent itemsets are those sets of data items that can be found always together more than a given number of occurrences in data. In other words, the goal of frequent itemsets mining is to determine which elements in a database (or any other data source) commonly appear together.

One scenario that is gaining a lot of attention of researchers is the data streams mining. Analyzing 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 hardware for data streams 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, the theoretical basis that support this research is presented. A review of state-of-the-art is addressed in section 3 while section 4 presents the methodological foundations of this research. The preliminary results are shown in section 5 while this paper is concluded in section 6. Also in section 6 the future works is drafted.

2 Theoretical basis

Let $I = \{i_1, i_2, .., i_n\}$ be a set of items:

Definition 1 (Itemset). *A itemset X is a set of items over I such $X = \{i_i, ..., i_k\} \subseteq I$.*

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

Definition 3 (Support). *The support of an itemset X is the number of transactions that contains X.*

An itemset is called $frequent$ if its support is no less than a given absolute minimal support threshold ϕ_{abs}, with $0 < \phi_{abs} \leq |D|$, while D is the mining database.

Definition 4 (Data streams). *A data stream is a continuous, unbounded and not necessarily ordered, real-time sequence of data items.*

Three characteristics appear in data streams: (1) Items in stream arrive continuously at a high rate (*Continuity*); (2) Items can be accessed and processed just once by the processing units in data streams (*Expiration*) and (3) The only assumption that we can make about bounds of streams is that the total number of data is unbounded and potentially infinite (*Infinity*).

Definition 5 (Window). *A window in a data stream is an excerpt of items that pertain to the stream.*

Windows can be created using one of this three approaches: (1)*Landmark window model*, (2)*Damped window model* and (3)*Sliding window model*. The Landmark Window Model employs some points (called landmark) to start recording where a transaction begins and ends. The support count of an itemset in this model is the number of transactions containing it between the landmark and the

current time. To distinguish between the oldest and new transactions a variation of this model was proposed and named Damped Window Model. Damped Window Model assigns different weights to transactions where the recent ones have weight near to 1, and older ones have weight near to 0. As time passes, the weight of each transaction will be degraded. The Sliding Window Model uses only the latest W transactions in the mining process. As the new transactions arrives, the old ones in the sliding windows are excluded. The use of this model impose a restriction: as some transactions will be excluded of the mining process, methods for finding expired transactions and for discounting the support count of the itemsets involved are required.

2.1 Reconfigurable Computing

Reconfigurable Hardware Computing is referred to the use of hardware devices in which the functionality of the logic gates is customizable at runtime, and FPGAs are the main exponent of this approach. 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 allows the creation of fully customized architectures, reducing cost and technological risks that are present in traditional circuits design.

Although there are other hardware development platforms for data streams mining (such as Graphics Processing Units, named GPUs), FPGAs are better suited. GPUs are graphic accelerators which are interfaced by the PCI port; while FPGAs can be interfaced by the PCI port, the USB port or the Ethernet connector (Ethernet interface is ideal for network stream analysis). Due to the high incoming rates of items in data streams, the processing such items must be done as fast as it can. FPGAs are great for real-time systems, where even 1ms of delay might be too long, and this capability are extremely valuable for mining data streams. GPUs are ideal for hybrid applications where some instructions must be accelerated while FPGAs can accelerate the whole process. Because of this, FPGA is better suited to be chosen as a development platform for accelerate frequent itemsets mining on data streams.

3 Algorithms review in hardware

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

Analyzing the revised literature it can be noticed that frequent itemsets mining on data streams using reconfigurable hardware is an interesting research area so, it is worth to propose new parallels algorithms to face such task. In

Table 1. Algorithms and architectures for frequent itemsets mining in data streams using FPGAs. DB stands for *Database*; Apr for *Apriori* and FPG for *FP-Growth*.

Title	**Based**	**Source**
An Architecture for Efficient Hardware Data Mining Using Reconfigurable Computing Systems. [2]	Apr	DB
Hardware Enhanced Mining for Association Rules. [5]	Apr	Stream
Hardware-Enhanced Association Rules Mining With Hashing and Pipelining. [12]	Apr	DB
Novel Strategies for Hardware Acceleration of Frequent Itemset Mining With the Apriori Algorithm. [11]	Apr	DB
Mining Association Rules with Systolic Trees.[9]	FPG	DB
A Reconfigurable Platform for Frequent Pattern Mining.[8]	FPG	DB
A Highly Parallel Algorithm for Frequent Itemset Mining. [6]	FPG	DB
Design and Analysis of a Reconfigurable Platform for Frequent Pattern Mining. [10]	FPG	DB
An FPGA-Based Acceleration for Frequent Itemset Mining. [13]	Eclat	DB
FPGA Acceleration for Intersection Computation in Frequent Itemset Mining. [7]	Eclat	DB

this task, there are three main approaches: algorithms that use Apriori as the starting point, algorithms that use FP-Growth and those that use Eclat.

The algorithms that mimic the Apriori-based schemes in hardware require loading the candidate itemsets and the database into the hardware. This strategy is limited by the capacity of the chosen platform: if the number of items to manage 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, and this is a very time consuming approach. In addition, several candidates itemsets and a large database may cause a bottleneck in the system. This issues are forbidden in data streams mining.

As well as Apriori-based algorithm, the FP-Growth-based algorithms need to download the mining database to FPGA. They also need two passes over the database except Mesa et al. [6] but this one still need to download the database to the hardware device. This is impractical in data stream mining scenario due to the Expiration restriction. Like others reviewed algorithms, authors focused their attention in better data structures rather than substantial theoretical contributions. As rule, FP-Growth based algorithm can handle a limited number of itemsets, less than 11 in the better cases which is inadequate for real-life applications. Nevertheless, those algorithms based on FP-Growth use the FP-Tree data structure which is very well suited for data stream mining applications.

Eclat-based algorithm uses the vertical database representation in order to save memory and processing time. It use the intersection of items to compute the support, and it is more efficient than hash-trees. All the Eclat-based imple-

mentations propose an hybrid approach, where the most consuming functions were download to hardware while software controls the execution flow and data structures. Due to the bandwidth limitations of used hardware devices, very large transactions must be segmented. In the reviewed papers, no segmentation strategies were reported. Although the vertical database representation allows to save memory and processing time, it is not compatible with the Expiration restriction. Also, the pruning strategy in Eclat is inefficient and introduces delays that affect the performance of the algorithms. This two issues make Eclat impractical to be used as a starting point for data stream mining algorithms.

4 Methodological foundations

4.1 Research problems

Modern applications generate huge data volumes in 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 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, software that analyzes such data streams can not process exhaustively all items. The supporting hardware and software are not capable to deal with such intense processing. Instead, commercial applications that mine data stream use an "approximate" processing approach. That is, they do not analyze all 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 applications that need intense processing requirements, e.g. intrusion detection systems or network analysis systems. In this kind of applications, the immediate data analysis and near-real-time response are extremely valuable. To fulfill these requirements is needed to propose new parallel algorithms running on high-performance computing devices such as FPGAs. FPGAs can perform tasks in a high parallel fashion, and this is very useful in data streams processing applications.

Frequent itemsets mining is one technique that is commonly used in data knowledge extraction and have been used with success in databases scenario. To mine frequent itemsets in data streams efficiently, an alternative would be to develop new parallel approaches that use custom hardware architectures. In the reviewed literature, there is only one architecture to mine frequent itemsets on data streams [5].

4.2 Aims and expected contributions

The general aim of this research work is: *To develop parallel methods for frequent itemsets mining in data streams that outperform the state-of-the-art algorithms for data streams analysis and that are suitable for being implemented in hardware-accelerated platforms. The proposed methods must outperform in one order of magnitude (at least) the state-of-art algorithms implemented in software.*

To fulfill the general aim, some specific aims were proposed: (1) To propose a flexible method for separating the incoming data stream into windows that it can be used by the support counting algorithm; (2) To adopt data structures 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 selected and the data structures adopted; (4) To obtain parallel hardware implementation of the algorithms mentioned above that can perform frequent itemsets mining at least 1 order of magnitude faster (without compromising effectively) than state-of-the-art software implementations.

As results of this research, the following contributions are expected: (1) A new method for frequent itemsets mining on data streams; (2) A design of parallel one-pass algorithms to mine frequent itemsets on data streams and (3) A custom hardware architecture that implements the proposed algorithms. This custom architecture will take advantage of inner parallelism provided by the hardware device used in its implementation.

5 Preliminary Results

After the literature was reviewed, the conclusion observed is that the selected window model should not be an issue: our method, and therefore, the hardware designs derived, must work fine regardless of the window model selected. So, the window model will be an input parameter.

The basic idea of the presented method is to develop a tree structure of processing units where the itemsets in data streams flow from the root node to leaf nodes. The tree structure presented is named *systolic tree* and each node has one child and one sibling. For leaf nodes, the child and sibling nodes are null nodes. In this structure, the child node contains, as a prefix, the itemset handled by its parent. Fig. 1 represents the systolic tree.

Using the Apriori property, which states that any subset of frequent itemset must be frequent [1] if a node is regarded as frequent then its parent is frequent too with equal or greater frequency counting. This property is specially useful for frequent itemset selection strategy.

The size (in number of nodes) of the systolic tree is determined by the capacity of the development platform. Assuming that the development platform contains enough computational resources, the size (in number of nodes) of the systolic tree will be $k = 2^n - 1$. The nodes in the systolic tree have its own processing logic, which it is presented in Algorithm 1. The systolic tree data structure presented in this paper implements a distributed control scheme: the

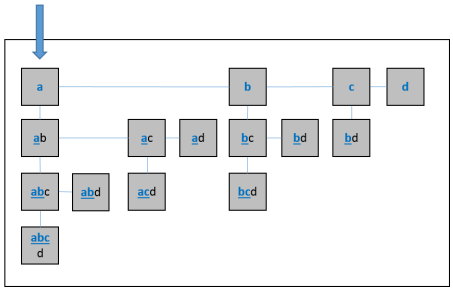

Fig. 1. Systolic tree data structure to mine frequent itemsets on data streams.

processing and control logic are distributed in each node of the systolic tree. This allows saving computational resources due to the logic reduction.

When the data stream arrives, each transaction is flowed into the systolic tree to determine the frequency of each item. Algorithm 1 depict the frequency counting scheme proposed. This algorithm will be executed in parallel in each node of the systolic tree. After the frequency of each itemset is calculated, those itemsets that can be regarded as frequent are determined using a backtracking strategy and the Apriori property.

The proposed method is designed to be implemented in a custom hardware architecture. To validate the concept introduced in this research, it was programmed sequentially in software using C# language over the .Net Framework platform.

As it was explained earlier, the systolic tree can process a limited number of items determined by capacity of the hardware device used. If the chosen development platform can hold a systolic tree with 1024 nodes, the maximum number of different items in the incoming transactions that it can be process will be 10. If the number of different items in the incoming transaction is greater, some itemsets will be not processed and therefore, the mining process will be approximate with no false positives produced. However, if the development platform can hold all the possibles itemsets, the mining process will be exact.

Some experiments were conducted and the pursued objectives were to verify the correct performance of the proposed algorithms and to measure how the systolic tree grows according to the length of incoming transactions. To accomplish these goals, MSNBC dataset from UCI repository [3] was used. This dataset is click-stream data that contains 989,818 sequences. The number of distinct item in this dataset is 17. The average number of itemsets per sequence is 13.33. The average number of distinct item per sequence is 5.33. In order to validate that the frequency counting computed by algorithm 1 is correct, it was assumed that the systolic tree can handle all possibles itemsets for used dataset. It is important to notice that for various selected support values, the conducted experiments demonstrate that the frequent itemsets detected by the proposed method and its frequency countingwas the same that the one obtained by the baseline FP-Growth.

Algorithm 1: Frequency counting.

Input: Transaction's window
Output: Systolic tree with the counting frequency of each itemset.

```
1  n_i ⟵ systolyc_tree.RootNode;
2  foreach itemset S_i in window_buffer do
3      Flush S_i into n_i;
4      if n_i.IsOccupied == false then
5          n_i.IsOccupied = true;
6          n_i.Label.Add(S_i[0]) n_i.Counter++;
7          S̃_i = S_i.Exclude(n_i.Item) if S̃_i.IsEmpty == false then
8              StartParallalelBlock:
9              n_i ⟵ n_i.ChildNode;
10             Flush S̃_i to n_i and go to step 4;
11             n_i ⟵ n_i.SiblingNode; Flush S̃_i to n_i and go to step 4;
12             EndParallelBlock;
13     else
14         if S_i.Contain(n_i.Label) == true then
15             n_i.Counter++;
16             S̃_i = S_i.Exclude(n_i.Item);
17             if S̃_i.IsEmpty == false then
18                 StartParallalelBlock:
19                 n_i ⟵ n_i.ChildNode; Flush S̃_i to n_i and go to step 4;
20                 n_i ⟵ n_i.SiblingNode; Flush S̃_i to n_i and go to step 4;
21                 EndParallelBlock;
22         else
23             n_i ⟵ n_i.SiblingNode; Flush S_i to n_i and go to step 4;
24 return systolic_tree;
```

Fig. 2 shows graphically the results obtained. Experiments show that the systolic tree grows exponentially (and therefore memory consumptions grow exponentially too) concerning to the length of the incoming transactions. This effect can be attenuated using larger hardware devices or using external memories, but it is still an issue to be taken in account. The processing time grows linear concerning of the length of the incoming transactions. These values are calculated for the sequential software implementation. The proposed method is designed to be implemented in parallel so the processing will be executed simultaneously, and after some initial time, the results will arrive continuously.

The software implementation of the proposed method pursuits the main objective of determining whether it is a valid solution for frequent itemset mining on data streams while in future works, hardware implementations will be developed. Experiments demonstrate that for different support values, the frequent itemsets and its frequency counting are the same that obtained by the baseline

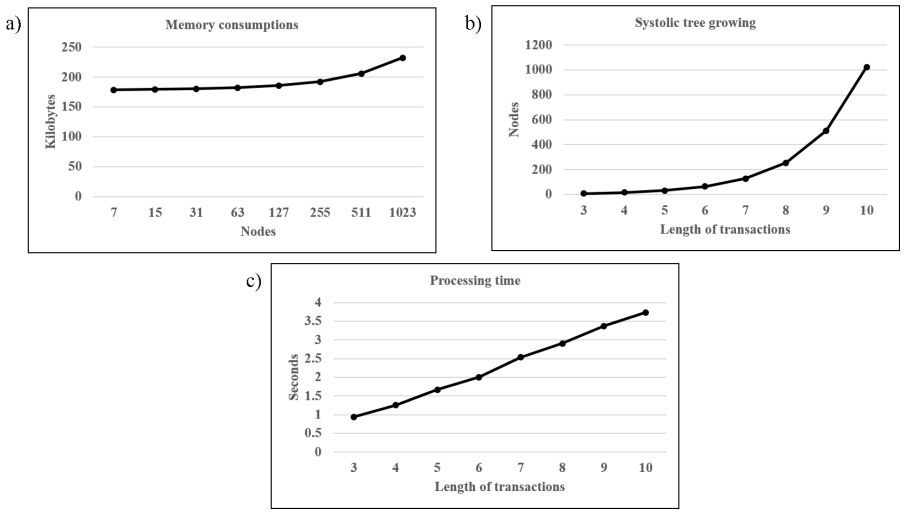

Fig. 2. Behavior of the proposed method while the length of the incoming transactions grows. a) Memory consumption graph. b) Systolic tree size graph and c) Processing time graph.

software. The length of the incoming transactions, and therefore the systolic tree size, can affect these results. In this case, not all of the frequent itemsets will be returned, but those itemsets that are regarded as frequents by the proposed method will be regarded as frequent with the same frequency counting by the baseline FP-Growth. In other words, if the available computing resources of the development platform selected can handle any length of the incoming transactions, the mining process will be exact. Otherwise, the mining process will be approximate with no false positives. The software implementation validates the correct functioning of the proposed method and allows to understand its functioning before implement it in hardware.

6 Conclusions and future work

Frequent itemset mining is a widely used Data Mining technique with outstanding results in database scenario. Data stream mining is a recent research field where frequent itemsets are introducing. Due to the continuity, expiration and infinity characteristic of data streams it is necessary to explore alternatives that allow to increase the efficiency of the mining process in such datasets. One alternative could be the design of parallel algorithms to be implemented in custom hardware architectures.

This paper introduce a new parallel method for frequent itemset mining in data streams which is designed to be implemented in a custom hardware architecture. The proposed method implements a distributed control logic among all processing nodes, and each node execute the same algorithm. Some experiments

were conducted, and it can be concluded that the proposed method correctly performs this task. When it is executed in a device with no resources restrictions then the exact mining process is performed. By the contrary, when restrictions are imposed, then the approximate mining process with no false positives is performed. From the experiments conducted it is derived that some adjustments must be done to the proposed method in order to save computational resources of the selected hardware device.

In future works, the implementation in hardware is mandatory. Also, a preprocessing strategy in order to determine 1-frequent itemsets which will be flowed into hardware architecture is the next step: this allow to optimize the nodes consumption in systolic tree. A segmentation database strategy that allow to handle larger datasets is an issue to deal with, and it is currently studying.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc. (1994)
2. Baker, Z., Prasanna, V.: An architecture for efficient hardware data mining using reconfigurable computing systems. In: Proc. of the 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (2006)
3. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`
4. Lin, C.H., Chiu, D.Y., Wu, Y.H., Chen, A.: Mining frequent itemsets from data streams with a time-sensitive sliding window. In: SDM (2005)
5. Liu, W.C., Liu, K.H., Chen, M.S.: Hardware enhanced mining for association rules. In: Proc. of the 10th Pacific-Asia Conf. on Advances in Knowl. Disc. and Data Mining (2006)
6. Mesa, A., Feregrino-Uribe, C., Cumplido, R., Hernandez-Palancar, J.: A highly parallel algorithm for frequent itemset mining. In: Advances in Pattern Recognition, LNCS, vol. 6256 (2010)
7. Shi, S., Qi, Y., Wang, Q.: Fpga acceleration for intersection computation in frequent itemset mining. In: Cyber-Enabled Distributed Computing and Knowl. Disc. (CyberC), 2013 Int. Conf. on (2013)
8. Sun, S., Steffen, M., Zambreno, J.: A reconfigurable platform for frequent pattern mining. In: Proc. of the 2008 Int. Conf. on Reconfigurable Computing and FPGAs (2008)
9. Sun, S., Zambreno, J.: Mining association rules with systolic trees. In: FPL (2008)
10. Sun, S., Zambreno, J.: Design and analysis of a reconfigurable platform for frequent pattern mining. IEEE Trans. on Parallel and Distributed Systems 22 (2011)
11. Thöni, D.W., Strey, A.: Novel strategies for hardware acceleration of frequent itemset mining with the apriori algorithm. In: 19th Int. Conf. on Field Programmable Logic and Applications (2009)
12. Wen, Y.H., Huang, J.W., Chen, M.S.: Hardware-enhanced association rule mining with hashing and pipelining. IEEE Trans. on Knowl. and Data Eng. 20(6) (Jun 2008)
13. Zhang, Y., Zhang, F., Jin, Z., Bakos, J.D.: An fpga-based accelerator for frequent itemset mining. ACM Trans. Reconfigurable Technol. Syst. 6(1) (May 2013)