A SURVEY ON FREQUENT ITEMSET MINING TECHNIQUES USING GPU

Ayushi M Patel, Dharmesh Bhalodiya
Computer Engineering Department
Silver Oak College of Engineering and Technology, Ahmedabad

***Abstract*—Frequent pattern mining is a discipline with many practical applications, where massive computational power and speed are required. Many state-of-the-art frequent pattern mining applications have inefficient solutions for both shared memory and multiprocessor systems due to problems with parallelism and memory. One of possible solutions to the trouble is the use of Graphics Processing Unit (GPU) in the organization along with modification of classical mining algorithms in such a manner, that the sequential part of the algorithm is run on the server and the parallel port on GPU.s. Here we present a survey of multi-core and GPU accelerated parallelization of the FIM algorithms**

***Index terms* – GPU , GP-Apriori , computing ,Parallel computing**

I. INTRODUCTION

Data mining is the process of discovering interesting, meaningful, and understandable patterns hidden in large data sets [1]. Now a day's organization collects sales data and this data is stored in form of transaction. Each transaction represents sales order. Each record stored in such database represents a transaction and attribute represent item parches by the client.

Let's consider an example of supermarket in which massive quantities of data continuously being taken in and laid in. They apply market basket analysis to analyze customer buying habits and discovering patterns that occur most frequently in the database. For example, 65 percent of clients who buy bread will also buy the butter. Association rule mining is the process of finding interesting relationship, patterns between the unrelated data in the transactional databases or data repositories. Apriori algorithm is used for finding association rules among items in market-basket data [2]. . Association rules have two main characteristics, i.e. minimum support and minimum confidence. Support is defined as a ratio of the number of transaction that the total number of transactions. Whereas Confidence is defined as the ratio of number of transactions which hold rule to number of transaction containing antecent. This mining process is split into two sub process. The first measure is obtaining those items which occurrences in the database or across the minimum support count or lower limit threshold. It is called frequent item-set. And the second step is for generating patterns from those frequent item-set with condition which satisfy the minimum confidence.
There are many application areas where item set mining is used such as retail business, bioinformatics and medicine, fraud detection and network intrusion detection and many more. The basic advantage of Apriori gives an efficient, frequent item set in transactional database as an output. Where as its dis-advantages are repeatedly scanning the transaction databases and get a big number of candidate item- position [1].

Apriori Introduction

Apriori employs an iterative approach known as point-wise search, where k-itemsets are used to explore (k+1) -itemsets. Let D be the market-basket database whose each row contains T Transactions tagged with unique identifier tied. here let me be the item set {I1, I2, I3, I4}.
If an item set contains k-items then it is called k-itemsets and all its subset satisfy the minimum support count then it is called Lk frequent itemset or large itemset. Atwo-step process is followed,(a) Join, self-join with previous frequent k-itemset and than create new candidate Ck+1 itemset. (b) Prune, remove the subset of current candidate itemsets which are not frequent in the previous measure. Below working of apriori algorithm is explained.

1. Scans all the transactions of database to find out candidate 1-itemset C1 to count the number of occurrences of each point.
2. Suppose that minimum confidence and minimum support count are given as min-conf and min-sup respectively.
3. Eliminate items from C1 whose count doesnot satisfy minsup threshold.Remaining1-items in C1is called L1.
4. To find the set of frequent 2-itemsets, L1 join L1 and create the new C2.n scan the entire database and calculate the number of times candidate 2-itemset appeared in the database.
5. Use the pruning in C2 and get L2.
6. In this way procedure step 2 to 5 is carried out until CK is empty or nil.

CUDA Programing

At the start of multicore CPUs and GPUs the processor chips have become parallel systems. Only the

focal ratio of the program will be increased if software exploits parallelism provided by the underlying multiprocessor architecture [5]. Therefore, there is great demand to design and develop the software that supports muthithreading,and each thread running concurrently on a processor, results in increasing the speed of the program dramatically. NVIDIA's graphics processing units (GPU) have evolved into a highly parallel, multithreaded, many-core processor (as depicted in Figure 1 (a)) with tremendous computational horsepower and very high memory bandwidth. GPUS have hundreds of processor cores and thousands of threads concurrently running on these substances. It relies on the massively multi-threaded SIMD (Single Instruction, Multiple-Data) architecture provided by GPUs.

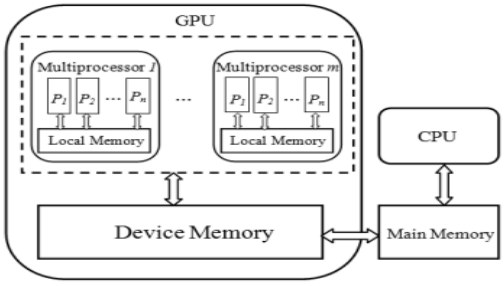

Fig.1 Architecture of Parallel Processor

CUDA stands for Compute Unified Device Architecture. It is a parallel programming paradigm released in 2007 by NVIDIA. It is practiced to acquire software for graphics processors and is used to prepare a sort of general purpose applications for GPUs that are extremely parallel in nature.

The CUDA parallel programming model is designed to surmount this challenge while maintaining a low learning curve for software engineers familiar with standard programming languages such as C. CUDA has some specific functions called kernels kernel can be a funtion or a full program invoked by CPU. IT is executed an N number of times in parallel on GPU by using N threads. Each thread that executes the kernel is presented a unique thread ID that is accessible within the meat through the built-in threadIdx variable.. CUDA grants shared memory and synchronization among the threads. CUDA threads may access data from multiple memory spaces during their performance. Each thread has private local memory. Then the number of threads per block, i.e., each thread block has shared memory visible to all threads of the block and with the same lifetime as the cube. .All the threads of blocks per grid have access to the same global memory.There are two additional read-only memory spaces accessible by all threads i.e. texture and constant memory spaces.Texture memory is read from kernels using the device functions described in Texture Functions. Reading data from texture or surface memory instead of global memory can have several performance benefits.Texture memory also offers different addressing modes, as well as data filtering, for some specific data formats.

Parallel Algorithms

Agrawal& Shafer [11] was staged the first parallel version of Aprior. Three different parallel versions of the Apriori method are given in [11]. Among these methods, the database is supposed to be distributed horizontally among the central processing units. The foremost method is called *Count Distribution (CD),* which is a straight-forward parallelization of Apriori. Each processor works out the partial backing of all candidates itemsets from its local database partition. At the end of each iteration, the processors exchange their partial supports to measure the global supports. The second method is called Data Distribution (DD), which partitions the candidate itemsets into disjoint sets and attributes them to different CPUs. In the DD method, each processor has to skim the full database (not only its local partition) in all iterations, to measure the global livelihood. Thusly, the DD method involves a high communication overhead. The *Candidate Distribution* algorithm follow the same strategy applied in *Data Distribution*, but it selectively replicates the dataset, the reason behind that each processor proceeds independently. The local component of the dataset is still scanned in every iteration. Among the three parallel versions of Apriori, the CD method is reported to perform the best. Many algorithms can use one of the above strategy to parallelize it. Like AprioriDP[9] was dynamic and triangle based method to find frequent2-itemset.

The rest of this paper is structured as follows. Part 2 describes a survey of multi-core and GPU accelerated parallelization of the FIM algorithms. Section 3 shows the comparison between those algorithms and last section shows the conclusion.

II. LITERATURE SURVEY

In [6],[7],[14] presented were three modified versions of the most basic Apriori algorithm, adapted for use with GPU:Apriori TBI, Apriori PBI and GPApriori.

2.1 PBI & TBI algorithm[6]

Wenbun Fang and Mian Lu.[6] group of authors proposed Apriori[10] and was first time addressed parallel version of FIM[4]. Two different approaches are *pure bitmap* and *trie-based bitmap*.Transactions and itemsets are encoded in bitmap and transfer in GPU but it will degrade the performance. Firstly, traditional approach (1) was used and nowadays vertical representation (2) is used and demonstrate speed factor which is higher than traditional

one. PBI-GPU is faster in dense dataset. TBI-GPU is better in sparse dataset. PBI stores itemsets in a bitmap structure and look up table is used to avoid counting the number of 1's in integers, so table stores the mapping of integer in its binary representation. **TBI** uses a tree structure to store itemsets and adopts co-processing scheme [6].

Item	TID
1,2,3,4	1
2,3,4	2
3,4	3
1,3,4	4

(1)

Item	Tidset	Bitset
1	1,4	1001
2	1,2	1100
3	1,2,3,4	1111
4	1,2,3,4	1111

(2)

Fig.2 dataset[6]

PBI-GPU vs TBI-GPU[12] comparison shows the effect of dfferent itemset representations - bitmap-based and trie-based. PBI-GPU and TBI-GPU invokes exactly the same support counting procedure on the GPU. The performance di®erence only comes from the candidate generation. The dense data set Chess has very few items (75 in total), hence the bitmap representation of itemsets for PBI-GPU is of small size. On the other hand, the sparse dataset Retail has many items (16469 in total), so PBI-GPU should process large bitmap of itemsets. Thus, the number of items determines the performance of PBI-GPU's candidate generation. Therefore, PBI-GPU outperforms TBI-GPU on denser data set, due to the smaller size of bitmap representation for itemsets, while TBI-GPU is better on the sparse dataset. [12].

2.2 GPAprioriAlgorithm[7]

Fan Zhang, Yan Zhang, and J. Bakos[7] proposed GPU accelerated traditional Apriori implementation which is known as GPApriori. It adopts the same methods, like Support Counting, Candidate Generation and Candidate Pruning. Other frequent itemset mining algorithms either use vertical representation or horizontal representation but in GPApriori, they have introduced new Bitset representation of complete database.Thebitset representation requires more memory space but it is more suitable for designing a parallel set join operation, which is better suited for GPU."Botwise and" operations performed for joining two transaction list.

In support counting process is based on a complete intersection in which, candidates are copied from main memory to graphic memory by host code, the GPU uses bitwise intersections on their vertical transaction lists, and are replicated back to main memory. Compared to the equivalent class clustering method, complete intersection adds computational complexity in order to reduce memory usage, but in the GPU cost of adding logic operations is lower. Carrying out of support counting in codes to ensure that coalesced memory access it aligned vertical list into 64 Byte. As bitset representation is used, it needs to enumerate the number of 1. The inbuilt code function pops (population count) is applied to calculate support count and store it in graphics memory in the form of vectors which are transferred to main memory. A parallel summation reduction algorithm [13] is used to tally all the support values recursively in its first constituent.

2.3 Frontier Expansion[14]

Fan Zhang, Yan Zhang, Jason D. Bakos[14] describe a new parallel Frequent Itemset Mining algorithm called "Frontier Expansion," an improved data-parallel algorithm derived from the Equivalent Class Clustering(Eclat) method, in which a partial breadth-first search is utilized. It has two main aims, firstly to finely parallelize Eclat's computational kernel for GPU acceleration and second is to achieve a dynamic tradeoff between performance and memory requirement, setting aside for a large data set to be processed with limited memory.

Algorithm utilizes Frontier stack for candidate generations. IT rapidly expands stack by consuming old, generating fresh and deleting infrequent candidates from stock for support counting, the frequency of new candidates is computed by intersecting the best vertical transaction list on the GPU. By visiting a big number of CUDA memory allocations and de-allocations, vertical list is generated and discarded. It allocates space for the upper limit number of vertical lists of GPU and stories free list addresses in a stack.a stack. When the program needs a free vertical list, it pops and returns an address stored.This method generates candidate of size K if candidates size of K-1 has the same prefix else they aren't in same class.

In frontier expansion during the expansion, the support for each of a set of new candidates are countered by GPU kernel and those meet the necessity of minimum support are forced back into stack. Candidates are sorted descendently by they support values and procedure is replicated until the heap is empty.

breadth first search will process more candidates in support counting phase and maximize the degree of data parallelism but the trade off is that the expansion requires a larger memory space.

2.4 Tree projection[8]

George Teodoro Nathan Mariano Wagner Meira Jr. Renato Ferreira[8] Describes Tree projection based frequent itemset mining algorithm. It is made up of core tree data structures, which is used to guide mining operations. Nodes assumes a lexicographic ordering

	PBI &TBI	GPApriori	Frontier Expansion	gpuDIC	Tree Projection
Kind of database	Bitmap	Bitmap	Bitmap	Bitmap	Vectors
Scalability on GPUs	No	Yes	Yes	No	Yes
BaseAlgorithm	Apriori	Apriori	Eclat and FPGrowth	Eclat	FPGrowth
Candidate Generation techniques	Bitwise (AND) &Prefix tree	Bitwise (AND)	Equivalent class (tree)	Not mention	lexicographic
Parallization strategy	Transaction	Transaction	Transaction	Transaction and candidate	Transaction and candidate

Table-1 Comparison of Algorithms[6],[7],[8],[14],[15]

among frequent itemsets in a database, which is built at run-time Depth represents the size or levels of itemset.Tree projection[8] employs a breadth-first strategy. Two parallelization methods for building tree are: (i) *the Transaction processing*, processing the transactions in parallel on the available hardware, each counting matrix of the nodes at level k- 1 is updated.; (ii) *the node level parallelism*, where each node being expanded is assigned to a different processor and updating its node matrix.

They demonstrate various locking techniques like node level, tree level to avoid race condition in the multi core CPU. If the node processing will create, load imbalance and needs more memory because matrices need to be updated concurrently in memory which is infeasible in many instances. On the other end If transaction processing is used, then execution time is controlled using an add count method that processes the transactions making interesting candidate for parallelism and so tree node expanded among parallel processor and synchronized accesses. Some other critical problem find is how to efficiently represent a lot of transactions on the GPU. And then they have introduced novel and compact vector base database representation.

2.5 Dynamic counting itemset [15]

Claudio Silvestri, Salvatore Orlando Universit`aCa' FoscariVenezia [15] group of authors proposed gpuDCI, parallel version of Dynamic Itemset Counting [8]. DCI is

a multi-strategy algorithm characterized by several phases, each exploiting a different strategy. The key idea, is all the subsets of a frequent set must be frequent. And so, the algorithm performs several iterations, starting with one Item patterns and increasing the size of searches, pattern at each iteration. DCI adopts scanning the database by transaction, using specific direct-count data structure to update the counters of the itemsets, After that when the number of surviving transactions and items allow vertical representation of the pruned dataset in memory. But later on DCIswitches to Intersection phase is more efficient as it has to deal with higher number of candidates than the Direct Count phase. It applies a bitwise data structure. Gpu-DCI examined *the transaction wise* and *candidate wise* strategies. Count phase would hinder the efficient exploitation of GPUs during this phase.Transaction wise parallelism was carried out by increasing stride with block id and thread id while in candidate parallelism stride with only thread id. Another novel approach is two level reductions, local reduction performed by each multiprocessor, data fetched from shared memory which is already exist in it. Global reduction may cause more penalty because data must be fetched from the counters which is residing in global memory.

III. COMPARISON

Apriori and Eclat iteratively generate k+1-sized frequent item sets by joining frequent k-sized item sets. This step is called candidate generation. After making each new set of candidates, the algorithm scans the transaction database to estimate the number of occurrences of each nominee. This amount is called support counting. The main difference between Apriori and Eclat is the manner they represent candidate and transaction data and the decree that they skim the tree structure that stores the candidates [7]. Apriori uses breath search technique and can be parallelized by parallel computation, however, results in extremely high memory usage [14]. Eclat uses depth search technique requires less memory if the utmost are small in number. FPGrowth has greater performance than

Apriori and Eclat but it requires high memory that prevents it being employed by large data sets. For high threshold value Apriori out performs FPGrowth[14].

Here we have chosen five different parameters and five algorithms identify the functionality. In kind of database parameter all GPU versions use bitmap representations because of heterogeneous environment, but tree projection based method vector representations. By comparing both the representations bitmap is complete matrix and therefore required more storage than a vector.

Second parameter shows algorithm can use more than one GPU. The first PBI & TBI and gpuDIC can't utilize the more than one GPU. Whereas for Frontier Expansion apply the old concept to parallelize, but even so it improves all over result in their experimentations.

Third parameter shows that from which algorithm they proposed the improvement in their clause. Frontier expansion and tree projection base proposed methods improve mainly Éclat and FPGrowth. Here GPApriori traditional apriori version. But here specific to DIC (dynamic itemset counting) they proposed parallel version gpuDIC.

Fourth parameter speaks tree base operation performed on CPU and Bitwise AND operation performed on GPU. As tree has unpredictable memory access and so GPU is not applicable. So candidate generation by tree like techniques that causes to be done on the CPU, in other word tree structure has to be managed by the CPU, then generated candidate move to the GPU memory.

The final parameter in the table is parallelization strategy. There are two main strategies, i.e. Transaction and candidate strategy to be used. gpuDIC and Tree projection utilize both the strategies and discuss that transaction wise parallelism is more suitable if we have enough resources. Whereas candidate wise parallelism gives unexpected outcome and also that it's not always outperform the transaction wise parallelism.

IV. CONCLUSION AND FUTURE WORK

Throughout the survey, we have spoken about various ways to solve frequent itemset mining problem. All the algorithms mentioned in the survey provide the same idea with a priori that can be date generation, support counting and candidate pruning.g. Implementing GPApriori algorithm and try to and reduce the items and increase its efficiency.

REFERENCES

[1]. R. Agrawal And R. Srikant. Fast Algorithms For Mining Association Rules.
IBM Research Report RJ9839, IBM Almaden Research Center, San Jose, California, June 1994.

[2]. Sheila A. Abaya, "Association Rule Mining Based On Apriori Algorithm In Minimizing Candidate Generation",In:International Journal Of Scientific & Engineering Research Volume 3, Issue 7, July-2012

[3]. M. J. ZakiAnd K. Gouda. Fast Vertical Mining Using Diffsets.In Proc. SIGKDD. 2003. P. 326-335

[4]. J. Han, H. Pei, And Y. Yin. Mining Frequent Patterns Without Candidate Generation.In SIGMOD. 2000. P. 1-12

[5]. R. Agrawal, T. Imielinski, And A. Swami. Mining Association Rules Between Set Of Items In Large Databases. In Proceeding Of The 1993 ACM SIGMOD International Conference On Management Of Data, Pages 207–216, May 1993

[6.] Mian Lu Xiangye Xiao Chi Kit Lam Philip Yang Bingsheng He Qiongluo Pedro V. Sander Wenbin Fang, Ka Keung Lau And Keyang.Parallel Data Mining On Graphics Processors. Technical Report HKUST-CS08-07, October 2008.

[7] Fan Zhang, Yan Zhang, And J. Bakos. Gpapriori: Gpu-Accelerated Frequent Itemset Mining. In 2011 IEEE International Conference On Cluster Computing (CLUSTER), Pages 590–594, 2011. Doi: 10.1109 /CLUSTER. 2011.61.

[8] George Teodoro Nathan Mariano Wagner Meira Jr. Renato Ferreira.TreeProjectionbased Frequent Itemset Mining On Multi-Core Cpus And Gpus. 22nd International Symposium On Computer Architecture And High Performance Computing, Pages 47 – 54, NOVEMBER 2010. Doi: DOI10.1109/SBAC-PAD.2010.15.

[9]Dharmeshbhalodia, K. M. Patel ,Chhaya Patel, An Efficient Way To Find Frequent Pattern With Dynamic Programming Approach ,NIRMA UNIVERSITY INTERNATIONAL CONFERENCE ON ENGINEERING, Nuicone-2013, 28-30 NOVEMBER, 2013

[10] NVIDIA CORPORATION, CUDA Programming Guide, Http://Developer.Nvidia.Com/Cuda

[11]Agrawal, R. And Shafer, J. 1996. Parallel Mining Of Association Rules.In IEEE Trans. On Knowledge And Data Engg., 8(6):962-969.

[12]Wenbin Fang, Mian Lu, Xiangye Xiao, Bingsheng He1, Qiongluo.Frequentitemset Mining On Graphics Processors.

[13] Nvidia. Data Parallel Algorithm In CUDA SDK Available From: Http://Developer.Download.Nvidia.Com.

[14]Yan Zhang Fan Zhang And Jason D. Bakos. Accelerating Frequent Itemset Mining On Graphics Processing Units. J Supercomput, Pages 94–117, NOVEMBER 2013. Doi: 10.1007/S11227-013-0887- X.

[15] Salvatore Orlando Universit A CaFoscariVenezia Claudio Silvestri. Exploiting GpusIn Frequent Itemset Mining. 20th Euromicro International Conference On Parallel, Distributed And Network-Based Processing, Pages 416–425, 2012. Doi: DOI10.1109/ PDP.2012.94.