RESEARCH ARTICLE OPEN ACCESS

Comparative Study of Frequent Itemset Mining Techniques on Graphics Processor

Dharmesh Bhalodiya[1], Prof. Chhaya Patel[2]
Computer Engineering Department, SOE, R K University, Gujarat, India

Abstract:
Frequent itemset mining (FIM) is a core area for many data mining applications as association rules computation, clustering and correlations, which has been comprehensively studied over the last decades. Furthermore, databases are becoming gradually larger, thus requiring a higher computing power to mine them in reasonable time. At the same time, the improvements in high performance computing platforms are transforming them into massively parallel environments equipped with multi-core processors, such as GPUs. Hence, fully operating these systems to perform itemset mining poses as a challenging and critical problems that addressed by various researcher. We present survey of multi-core and GPU accelerated parallelization of the FIM algorithms

I. Introduction

Data mining also called as Knowledge Discovery in Databases (KDD) [1]. Now a day many of the organization collect sales data. This data is stored in form of transaction, so each transaction represent sale order. In such database each record represent transaction and attribute represent item parches by customer. Now a day many of the organization collect sales data. This data is stored in form of transaction, so each transaction represent sale order. In such database each record represent transaction and attribute represent item parches by customer.

Take an example of super market. In that, each transaction is collected and after getting large amount of data. They apply market basket analysis and find out the pattern. The discovered patterns are set of items that are most frequent in database. Like 60 percent of people who buy bread also buy the butter. Decision making person use this detail for identify the customer buying habits.

Association rule mining is help in finding relationship among the set of items in all transactions. Apriory algorithm used for discovering association rules between items in market-basket data [2]. Association rule mining require two predefined values, those values are minimum confidence and minimum support. This mining process is divide into two sub process. First one for finding those items which occurrences in database across the minimum threshold or minimum support count. That is call frequent items set or large items set. And second one for generating of rules from frequent items set with condition is that it satisfy the minimum confidence.

Apriori Introduction

Let D be the market-basket database, where each row contains T Transactions. Transactions tagged with unique identifier Tid. Now let I be the item set { I1, I2, I3} If an item set contain k-item then it called k-itemset, and all subset of k-itemset satisfy the minimum support count then it's called Lk frequent itemset or large itemset. This algorithm need two basic steps are (a) Join, self-join with previous frequent k-itemset and create new candidate Ck+1 itemset. (b) Prune, filter from the current candidate itemset whose subset is not frequent in previous step.Below step explain the working of Apriory algorithm.

1. Assume that minimum support count and minimum confidence are given as min-sup and min-conf respectively
2. Scan the entire database and find out candidate 1-itemset C1 along with occurrence count. That is number of times each item appeared in database.
3. From C1 eliminate those items which count is not satisfy minsup threshold. Remaining 1-items in C1 which called L1.
4. L1 join L1, and create new C2, again scan the database and calculate number of times candidate 2-itemset appeared in database.
5. Apply the pruning in C2 and we get the L2.
6. As this way iteratively step 2 to 5 is carried out until the CK is null

CUDA Programing

At the start of multicore CPUs and GPUs the processor chips have become parallel systems. But speed of the program will be increased if software exploits parallelism provided by the underlying multiprocessor architecture [3]. Hence there is a big need to design and develop the software so that it uses multithreading, each thread running concurrently on a processor, potentially increasing the speed of the program dramatically. To develop such a scalable parallel applications, a parallel programming model is required that supports parallel multicore programming environment. NVIDIA's graphics processing units (GPUs) are very powerful and highly parallel. GPUs have hundreds of processor cores and thousands of threads running concurrently on these cores, thus because of intensive computing power they are much faster than the CPU.

CUDA stands for Compute Unified Device Architecture. It is a parallel programming paradigm released in 2007 by NVIDIA. It is used to develop software for graphics processors and is used to develop a variety of general purpose applications for GPUs that are highly parallel in nature and run on hundreds of GPU's processor cores.

CUDA uses a language that is very similar to C language and has a high learning curve. It has some extensions to that language to use the GPU-specific features that include new API calls, and some new type qualifiers that apply to functions and variables. CUDA has some specific functions, called kernels. A kernel can be a function or a full program invoked by the CPU. It is executed N number of times in parallel on GPU by using N number of threads. CUDA also provides shared memory and synchronization among threads.

Parallel Algorithms

Agrawal & Shafer [9] was presented first parallel version of Apriori. They implemented three different parallelization of Apriori on a distributed-memory machine (IBM SP2).The *Count Distribution* algorithm is a straight-forward parallel strategy of Apriori. Each processor has portion of full dataset and generates the partial sum (support) of all candidate itemsets from its portion of database partition. At the end global support is calculated by collecting all partial support from each processor. The *Data Distribution* algorithm partitions the candidates into disjoint sets, which are forwarded to each processor. To calculate the global support, each processor must scan the entire database from its local partition as well as from all other partition in all iterations. Thus it suffers from huge 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 portion of dataset is still scanned in every iteration. *Count Distribution* was shown to have superior performance among these three algorithms. Many algorithms can utilize one of above strategy to parallelize it. Like AprioriDP [10] was dynamic and triangle base method to find frequent 2-itemset.

Next section describe the brief summary of parallel algorithms implemented in NVIDIA architecture with the use of CUDA. Section 3 employ the detail comparison of CUDA base Apriori algorithms.

II. Related Work

Wenbin Fang and Mian Lu. [4]

These group of authors proposed Apriori[2] and it was first time addressed parallel version of FIM[4]. They have described two different approach, pure bitmap and trie-based bitmap. Transactions and items are coded in bitmap and then transfer in GPU memory. These may degrade the performance, because that bitmap size are larger then compare to[8]

Figure (a) shows traditional database representation. In figure (b) tidset is vertical database representation that used in many FIM algorithms and demonstrate speed factor is higher than traditional one. But in GPU tidset is not coalesced access and unpredictable memory read so it lead to poor performance. In contrast bitset is complete coalesced access. Below figure shows the coalesced access.

Item	T_{ID}
1,2,3,4	1
2,3,4	2
3,4	3
1,3,4	4

(a)

Item	tidset	Bitset
1	1,4	1001
2	1,2	1100
3	1,2,3,4	1111
4	1,2,3.4	1111

(b)

Figure 1 Data set

Fan Zhang, Yan Zhang, Jason D. Bakos[5]

They proposed GPU accelerated traditional Apriori implementation and named as GPApriori. It follows same methods of functionality, like Candidates Generation, Support Counting and Candidate Pruning. Other state of art of frequent itemset mining algorithms use either horizontal representation or vertical representation but in GPApriori, They introduced new Bitset representation of complete database.

	PBI & TBI [4]	**GPApriori[5]**	**Tree Projection[7]**	**gpuDIC[8]**	**Frontier Expansion[6]**
Parallelization strategy	Transaction	Transaction	Transaction and candidate	Transaction and candidate	Transaction
Candidate generation techniques	Bitwise (AND) &Prefix tree	Bitwise (AND)	lexicographic tree	Not mention	equivalent class(tree)
Kind of database	Bitmap	Bitmap	Vectors	Bitmaps	Bitmap
Scalability on GPUs	No	Yes	Yes	No	Yes
Base Algorithm	Apriori	Apriori	FPGrowth	DIC	Éclat and FPGrowth

Table 1: Comparison between various Techniques

For support counting process and to ensure that coalesced memory access it aligned vertical list into 64 Byte. As bitset representation is used it need to count number of 1. The in-built cuda function is used popc (Population Count) as these way they calculate the support count and store it into vector, that vector forwarded to CPU to generate candidate from that vector.

The same Group of author proposed "Frontier Expansion"[6] derived from Éclat. GPApriori is accelerated version of Apriori but Frontier Expansion accelerate more advance apriori such as Éclat and FPGrowth. It utilize Frontier stack for Candidate Generations with the help of Equivalent Class method. This method generates candidate of size K if candidates size of K-1 have same prefix else they arenot in same class. They have also proposed Producerconsumer model for support counting on multiple GPUs. The idea behind is that producer thread

Separate each equivalent class from Frontier stack and store it into its buffer, and consumer thread load equivalent classes at time from producer buffer than process it and store classes into producer buffer.

George Teodoro Nathan Mariano Wagner Meira Jr. Renato Ferreira[7]

They have proposed Tree Projection base frequent itemset mining algorithm. Tree Projection is made up from core tree data structure. Its nodes contain lexicographic ordered item from database and levels represent the size of itemset. They demonstrate the implementation in two different architecture and those are multi core CPU with sheared memory and GPUs. In multi core CPU they also discuss about various locking techniques like tree level, node level and call level, to avoid race conditions. They have mention two parallel strategies to carry out the FIM, those are (i) transaction wise and (ii) node level wise. According to their discussion if we apply node level parallelism then we cannot fully utilize the massive power of gpu because the load on processors are imbalance. In contrast they implement transaction wise parallelism, so tree nodes are shared among processors and synchronized accesses.

They have also introduce the novel and compact vector base database representation. Each transaction length is stored followed by its item. Additionally another vector used to map the starting index of the transactions. They addressed the issue of using bitset, which is complete matrix of transactions m v/s items n, require m x n, which is large amount of memory in sparse databases compare to proposed vector base dataset required.

Claudio Silvestri, Salvatore Orlando Universit`a Ca' Foscari Venezia[8]

In this article authors proposed gpuDIC, parallel version of Dynamic Itemset Counting on GPU [8, 9]. DIC is state-of-the-art FIM algorithm, article demonstrate the DIC outperform apriori [1] and FP-Growth [10] too. These group of author also examine the two parallel strategies (i) transaction wise and (ii) candidate wise parallelism. Transaction wise parallelism was carried out by increasing stride with

block id and thread id where in candidate parallelism stride with only thread id. Another novel approach is two level reductions, local reduction performed by each multiprocessor, data fetched from sheared memory which is already exist in it. Global reduction may cause more penalty because data must be fetch from counters which is reside in global memory. As a part of experiment they have used five different dataset and three different parameters was itemset length, transaction size and number of multiprocessors.

III. Comparison

We have taken five parameters and five algorithm to identify the overall functionality and strategy used behind individual. First parameter shows in table is parallelization strategy. There are two main strategies to be applied by various researcher. Tree projection and gpuDIC 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.

Second parameter tell us that from which techniques they generate the candidate. Bitwise AND operation performed on GPU and tree base operation performed on CPU. Because tree has unpredictable memory access and such in situation GPU is not applicable. So tree structure has to be handle by CPU, in other word candidate generation by tree like techniques that has to be performed on CPU, than generated candidate move to the GPU memory.

Most important parameter we have consider is that type of database used during their experiment. Because in such heterogeneous environment we have to utilize the maximum power of massively parallel computing hardware. Here all GPU versions used bitmap representation because of complete coalesced memory access and with the use of bitmap they gain the speed up in higher order of magnitude. GpuDIC use bitmaps according to processor type (x86 or x64). But tree projection based method introduce another way of representation is vectors. If we compare with vector and bitmap, bitmap need more memory space compare to vector. Because bitmap is complete matrix of M x N where M is number of items present in database and N is total number of transaction in database. Now vector is AlterNet way to represent the database in GPU kernel function. Actually theyrequire two vectors one for transaction index and second for items present in database.

Next parameter shows whether proposed authors algorithm can utilize more than one GPU. The first GP-GPU FIM and gpuDIC not utilize the more than one GPU. Whereas for Frontier Expansion apply old concept to parallelize, that is producer and consumer model. But still it improves all over result in their experiments.

Last parameter show that from which algorithm they proposed the improvement in their article, here GPApriori traditional apriori version. And frontier expansion and tree projection base proposed methods improve mainly Éclat and FPGrowth. But here specific to DIC (dynamic itemset counting) they proposed parallel version gpuDIC.

IV. Conclusions

In this survey, we have discuss that frequent itemset mining problem can be solved in many ways. The most widely used approach in current FIM is bitmap dataset. However, the work presented in this survey show the thriving efforts to improve over apriori since as it was demonstrated. All the algorithm present in this survey share the same idea with apriori that candidate generation, support counting and candidate pruning. Moreover their aims to demonstrate that GPUs can compute very large dataset in least time compare to CPU.

References

[1.] Jiawei Han and Micheline Kamber. Data mining concepts and techniques. Second Edition, morgan Kaufmann Publications, 2006.

[2.] NVIDIA CORPORATION, CUDA Programming Guide, http://developer.nvidia.com/cuda

[3.] 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.

[4.] Mian Lu Xiangye Xiao Chi Kit Lam Philip Yang Bingsheng He Qiong Luo Pedro V. Sander Wenbin Fang, Ka Keung Lau and Ke Yang.Parallel data mining on graphics processors. Technical Report HKUST-CS08-07, October 2008.

[5.] 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.

[6.] 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.

[7.] George Teodoro Nathan Mariano Wagner Meira Jr. Renato Ferreira. Tree projectionbased 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.

[8.] Salvatore Orlando Universit a Ca Foscari Venezia Claudio Silvestri. Exploiting gpus in frequent itemset mining. 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pages 416–425, 2012. doi: DOI10.1109/ PDP.2012.94.

[9.] Agrawal, R., and Shafer, J. 1996. Parallel mining of association rules. In IEEE Trans. on Knowledge and Data Engg., 8(6):962–969.

[10.] Dharmesh Bhalodia, 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

[11.] Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. kdci: a multi-strategy algorithm for mining frequent sets. In FIMI Workshop, 2003.

[12.] Salvatore Orlando, Paolo Palmerini, Raffaele Perego, and Fabrizio Silvestri. Adaptive and resource-aware mining of frequent sets. In IEEE ICDM, pages 338– 345, 2002.

[13.] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate genera- tion: A frequent-pattern tree approach. Data Min. Knowl. Discov., 8(1), 2004.