IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 21, Issue 1, Ver. I (Jan - Feb 2019), PP 16-25
www.iosrjournals.org

Mining Frequent Patterns with Optimized Candidate Representation on GPU using Parallel Eclat Algorithm

Janki Kansagra[1], Dr. Nilesh Kalani[2], Dr. Paresh Tanna[3]
[1]*Computer Engineering Department, School of Engineering, RK University*
[2]*Dean and Director, School of Engineering, RK University*
[3]*Computer Engineering Department, School of Engineering, RK University*
Corresponding Author: Janki Kansagra

Abstract: *Frequent itemset mining is one of the important aspects of association rule mining. The primary algorithm based for frequent itemset mining is mostly based on CPU and they generated a large set of items that are required to be kept in memory all the time while processing. In this dissertation thesis, we designed a parallel Eclat algorithm. The algorithm will run on GPU and perform the task of the frequentitemset mining in parallel. The proposed algorithm also uses the optimized candidate representation and the frequent item sets generated are stored in cache memory and are fetched directly from the cache memory. The proposed algorithm runs in parallel and also uses the optimized candidate representation and thus provides better performance than the classical eclat algorithm. Thus, the proposed algorithm runs much faster than the classical eclat algorithm and has better performance than classical eclat algorithm in terms of memory and time.*

Keywords: *FIM(Frequent Itemset Mining), Equivalent Class, Candidate sets, support count, Eclat.*

Date of Submission: 26-12-2018 Date of acceptance: 11-01-2019

I. Introduction

With the advancement in the storage and data capturing technology, many organizations have started storing their data. And created an ultra large database to store their scientific data. The advancement in processors now lead a new way to process the stored scientific data and derived some meaningful information from that data and use that data for decision making in business. Since last few years, the amount of data is increasing rapidly. It is very important to analyze the huge amount of data and generates meaningful information from the data. which can be used to draw some conclusions[1].

1.1 The process of extracting data

The process of extracting meaningful data from large data like a database, data workhouse is called association mining[2]. One of the very important aspects of association rule mining is to find frequent itemsets and find a relationship between various data variable and use this derived relationship for further analysis or conclusions. The process of association rule mining consists of two steps, First, one is finding item sets that have support count greater than minimum support count and the second step is identifying association rules among the frequent itemsets[1].

1.2 The main terminologies used in association rule mining are given below

1. ***Transaction database***: Transaction database is a set of transactions.
2. ***Item set***: An item set contains a list of items.
3. ***Support***: Support count specifies the number of transactions that contain item x and item y.

$$Support(x,y) = \frac{no.of\ transaction\ that\ contain\ x\ and\ y}{total\ no.of\ transactions.}$$

4. ***Confidence***: Confidence specifies to what extent the association rules generated are true. It is defined as the ratio of the numbers of transactions that contain an item x and number transactions that contain item x and item y

$$Confidence(x \rightarrow y) = \frac{no.of\ transactions\ that\ contain\ x}{total\ no.of\ transactions\ that\ contain\ x\ and\ y}.$$

This paper describes an improved version of the Eclat algorithm that runs in parallel on GPU and also uses the optimized candidate representation and thus generates the same result in less time compared to the classical Eclat algorithm. The first algorithm for frequent itemsets mining was Apriori algorithm that was

discovered in 1994. But the Apriori algorithm has disadvantages, one of the disadvantages was the entire database needs to be present in memory for processing and the second disadvantage was that it required many database scans. The first disadvantage remained in all the algorithms including FP-tree, Apriori, Eclat etc...[3]

II. Introduction to Eclat Algorithm

The Eclat algorithm[4] developed by Zaki is one of the serial algorithms to find frequent itemset from the given dataset. Eclat uses depth first search approach, unlike other algorithms that use the breadth first search approach. The éclat algorithm is a recursive algorithm that find frequent itemset from a given transaction database.

Initially, the Eclat algorithm finds single Tid-sets and find their support count and those items whose support count is greater than minimum support count is added to set P_x. And then we recursively go on intersecting Mindset and generating frequent itemsets and storing then in P_x. As we said that the Eclat algorithm uses depth first search, so the algorithm recursively finds all the frequent itemset in the branch of a particular item as shown in figure given below[4].

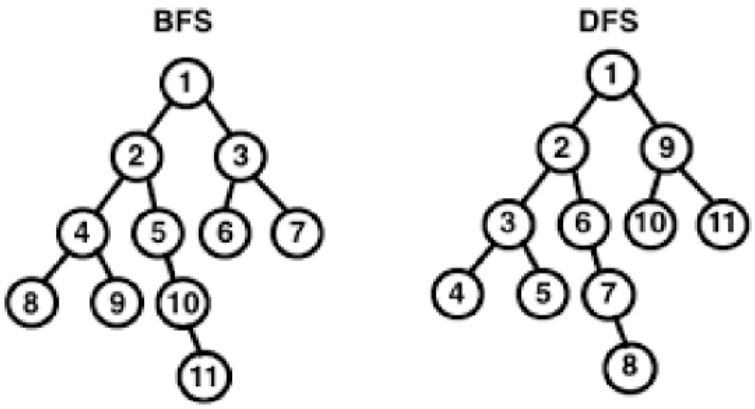

Figure 2.1: *Difference between DFS and BFS*

The Eclat algorithm takes the transaction database and a minimum support count as input and generates the frequent k-itemsets and then association rules are generated from the frequent item-sets using minimum confidence value. The transaction database is a set of items that are purchased together. Minimum support count is the minimum value of support count that candidate sets must have to be considered as frequent item-sets. The support count is the ratio of a number of transactions that have item x and item y together and the total number of transactions present in the transaction database.

$$\text{Support (x,y)} = \frac{\text{number of transactions that have items x and item y}}{\text{total number of transactions in the transaction database}}$$

The minimum confidence is the minimum value of confidence that item-sets must have to form the association rules. The confidence value indicates to what extent the rules derived from association mining are true. The confidence value is the ratio of the number of transactions that contain x and the number of transactions that contain both x and y.

$$\text{Confidence x} \rightarrow \text{y} = \frac{\text{number of transactions that contain x}}{\text{number of transactions that contain both x and item y}}$$

2.1 An Example of Eclat Algorithm

Let's consider one example transaction database that is as shown in Table 2.1 and Table 2.2 shows a vertical representation of transaction database shown in Table 2.1.

Table 2.1: *Sample Transaction Database*

TID	Itemset
T1	I1, I2, I3
T2	I3
T3	I2, I5
T4	I2, I3, I4
T5	I1, I2, I3
T6	I1, I4
T7	I2, I6
T8	I1, I7
T9	I1, I4

Table 2.2: *Vertical Data Representation*

	T1	T2	T3	T4	T5	T6	T7	T8	T9
I1	1	0	0	0	1	1	0	1	1
I2	1	0	1	1	1	0	1	0	0
I3	0	1	0	1	1	0	0	0	0
I4	1	0	0	1	0	1	0	0	1
I5	0	0	1	0	0	0	0	0	0
I6	0	0	0	0	0	0	1	0	0

Table 2.3 shows TID set representation of transaction database shown in TABLE 2.1 and TABLE 2.4 Shows support count for individual items of transaction database shown in TABLE 2.1.

Table 2.3: *TIDset Representation*

Items	TID sets
I1	1000110110
I2	1011101000
I3	0101100001
I4	1001010010
I5	0010000001
I6	0000001000
I7	0000000100

Table 2.4: *Support Count for 1-Itemsets*

Items	Support Count
I1	5
I2	5
I3	4
I4	2
I5	3
I6	1
I7	1

Let's assume that the minimum support count required for the itemset to be frequent is greater than 3. Then we need to remove those items whose support count is less than or equal to 3. So, we remove items I4, I5, I6, and I7 as their support counts are less than or equal to 3. Now let's generate 2-itemset candidate sets.

Table 2.5: *Support Count for 2-Itemsets*

Candidate Item-sets	Support Count
I1, I2	2
I1, I3	1
I2, I3	2

None of the 2-itemset candidate sets are frequent as none of them have support count greater than 3. So, the algorithm stops here, and the frequent items are I1, I2, and I3.

2.2 Advantages of Eclat Algorithm[4]

- The depth first approach reduces the memory requirement.
- It is faster than Apriori algorithm.
- No need to scan the database to find the support count of k+1 item-sets for, k less than 1.

2.3 Disadvantages of Eclat Algorithm[4]

- The TID sets are too large so manipulation of TID-sets is expensive.

III. Proposed Work

From here we will refer to the Proposed algorithm as Parallel Eclat Algorithm. The disadvantage of the classical algorithm is that as the transaction size is very large the calculation of frequent itemset is very expensive. So, in this dissertation the Parallel Eclat Algorithm divides the Tid'sinto a number of blocks and threads and calculation is done in parallel and thus this approach solves the problem faced in the Classical Eclat Algorithm.

The Parallel Eclat Algorithm also uses the vertical bit representation of the transactions same as the Classical Eclat algorithm because the vertical representation of the transaction database is very simple, and it requires only a single database scan to create the bit vector. The bit vector is reusable and can be used throughout the algorithm. Thus, improving the Performance of the algorithm than Apriori and FP-Growth.

The vertical representation[7] places 1 in the TID set if item I_i is present in the transaction T_i and places 0 if item I_i is not present in the transaction T_i. The advantage of using vertical bit set representation is that it is easy and well compatible with support count calculation and is much more efficient in terms of memory requirement than horizontal representation. Table 3.2 shows the vertical bit set representation for the sample transaction database shown in Table 3.1.

Table 3.1: *Sample Transaction DatabaseTable 3.2: Vertical bitset representation of Table 3.1*

Items	Transactions
T1	I1, I2
T2	I2, I3
T3	I1, I3
T4	I1, I2, I3

Items	TID set
I1	1 0 1 1
I2	1 1 0 1
I3	0 1 1 1

The Parallel Eclat Algorithm calculates the k+1 candidate item-sets for k>1 by the intersection operation between two items[1]. The candidate generation using intersection operation is shown in Figure 3.1.

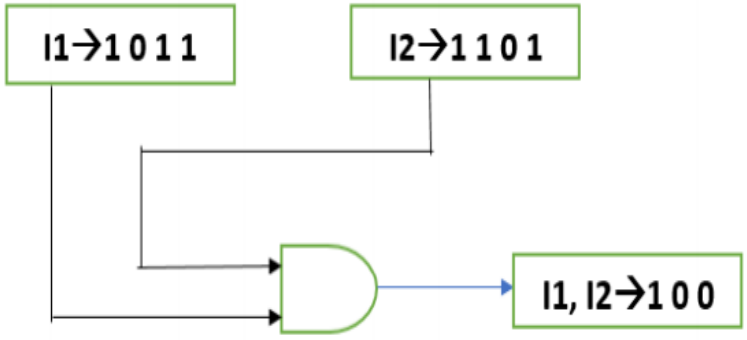

Figure 3.1: *Support counting using Intersection operation*

The support count is calculated by counting the number of 1's in the output of the intersection operation between TID sets of two or more items TID sets. For, example as shown in figure the support count of item-sets I1 and I2 from the transaction database shown in table 6.1 is calculated by performing intersection operation between TID sets of Item I1 and I2 the result of intersection is 1001 which contains two 1's so the support count of 2-itemsets I1 and I2 is 2.

The Parallel Eclat Algorithm stores the support count of the frequent item-sets i.e. item-sets whose support count is greater than the minimum support count in the cache memory. The advantage of storing the support count of frequent item-sets in the cache memory is that it can be accessed speedily compared to the main memory.

The Parallel Eclat Algorithm uses the parallel reduction operation to find intersection of two or more items TID sets and calculation of support count of k+1 item-sets where k>1. Suppose a transaction database has 32 transactions and a block of GPU contains 16 threads, we require two blocks in GPU to store the TID sets of Items. The Parallel Reduction is shown in Figure 3.2. It can be seen from Figure 3.2 that the long TID sets are divided into a number of blocks and threads thus overcoming the limitation of the Classical Eclat algorithm.

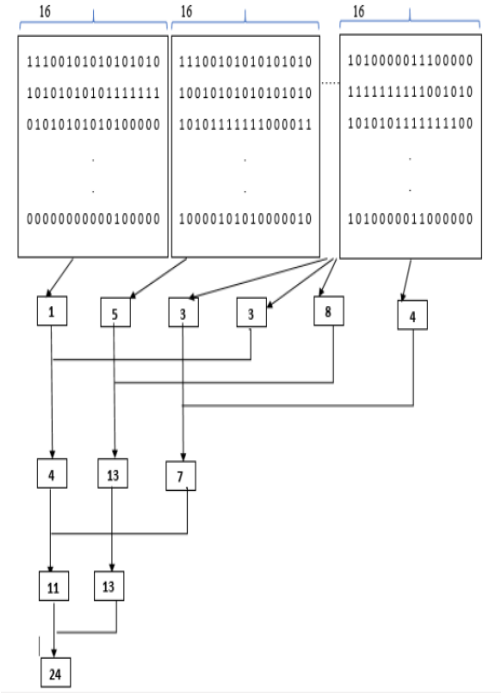

Figure 3.2 *Parallel Reduction*

The Parallel Eclat Algorithm also uses the optimized candidate representation to improve the performance further. The general approach of the Equivalent class is that 1,2,3 and 1,2,3,4 belong to the same equivalent class. And the level is k-1 but this thesis extends the level to k-n that is the item-sets 1,2 and 1,2,3,4 belong to the same equivalent class because the support count and the TID-sets of the frequent item-sets are stored in cache memory. Thus, reducing the time required in calculating intersection operation further for the newly generated candidate sets.

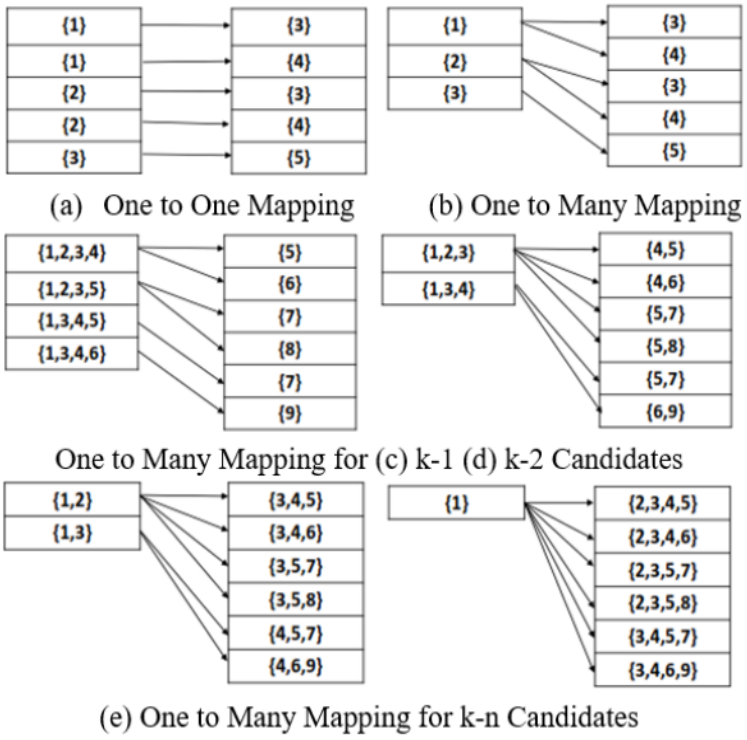

Figure 3.3: *k-n Optimized Candidate Representation*

3.1 Algorithm

The Algorithm for Intersection operation and parallel sum reduction is shown below.

Algorithm: Parallel Eclat
Algorithm: intersect()
Input: Tid of itemsets, size of block, min_support
Output: support count of candidate itemset stored in cache

```
__ shared__ int cache_result[];
int temp = 0;
int tid = threadIdx.x + blockDim.x * blockIdx.x;
while (tid < size)
do
c[tid]=a[tid] & b[tid]
temp += NumberOfSetBits_k(c[tid])
tid += blockDim.x * gridDim.x
done
cache_result[threadIdx.x] = temp
__syncthreads()
int i = blockDim.x / 2
while (i != 0)
do
if (threadIdx.x< i)
then
cache_result[threadIdx.x] += cache_result[i + threadIdx.x];
endif
__syncthreads()
i /= 2
done
if (threadIdx.x == 0)
then
support[blockIdx.x] = cache_result[0]
endif
```

IV. Experiment and Analysis

Wecarried our experiments on Unix Operating system with 8 GB RAM and GeForce 940MX and Driver Version 384 and Nvidia Cuda Version 9.2

4.1 Performance Analysis

Figure 4.1 demonstrates the performance result with respect to the minimum support count ratio on the real Mushroom dataset. There are 8124 data records or aff airs and the average transaction length is 45. The Parallel Eclat Algorithm provides better performance than Classical Eclat algorithm for very less minimum support count value and the improvement can be seen from the Fig. 4.1.

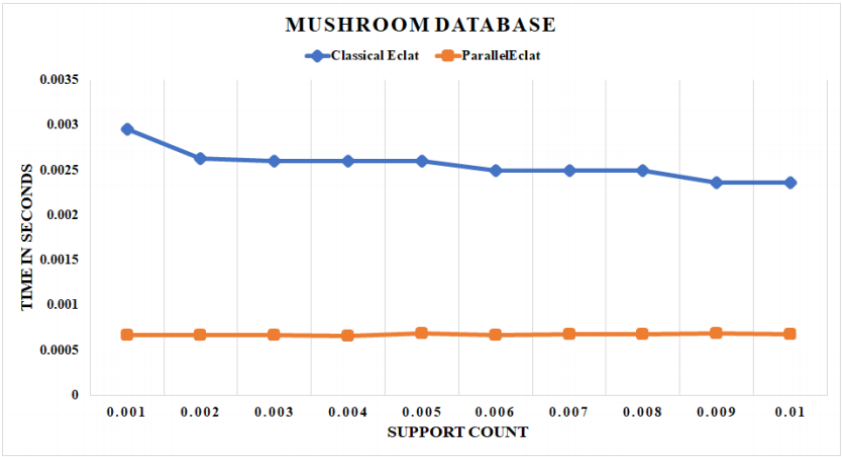

Figure 4.1: *Time Analysis for Mushroom Dataset*

It can be seen from Fig. 4.1 that the Parallel Eclat Algorithm requires less time than Classical Eclat Algorithm for very less value of minimum support count. Fig. 4.2 shows the results for the higher value of support count for the Mushroom database. It can be clearly seen from the figure that the results of Parallel Eclat Algorithm are better than Classical Eclat Algorithm.

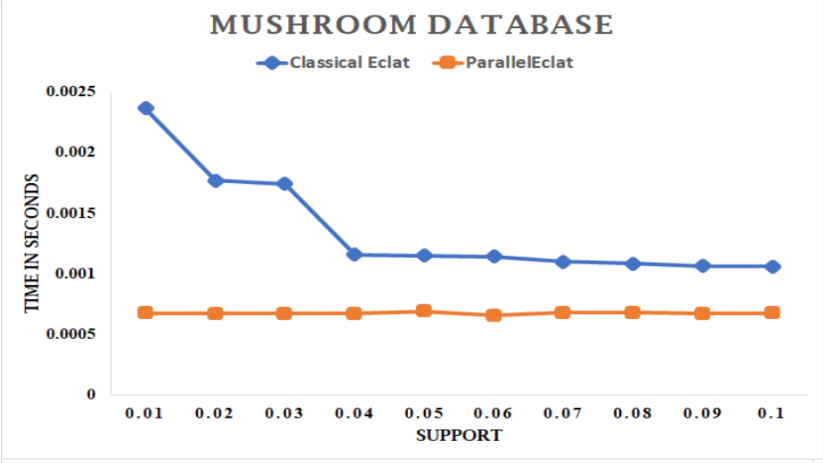

Figure 4.2: *Time Analysis for Mushroom Dataset*

We carried our tests on the second database that is the online retail database that has 514909 transactions in the transaction database and the average transaction size is 33. Time Analysis for both the Classical Eclat and Parallel Eclat Algorithm for support count 0.001 to 0.1 are shown in Fig. 4.3 and Fig. 4.4. It can be very clearly seen from the figures that the Parallel Eclat Algorithm has better performance for less value of minimum support and large transaction database size.

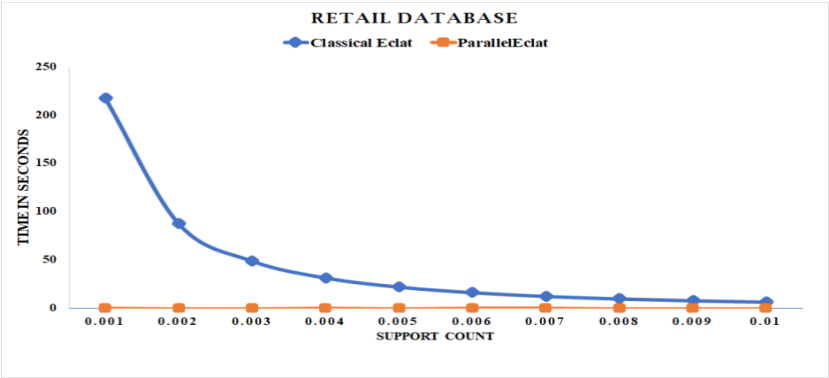

Figure 4.3: *Time Analysis for Retail Dataset*

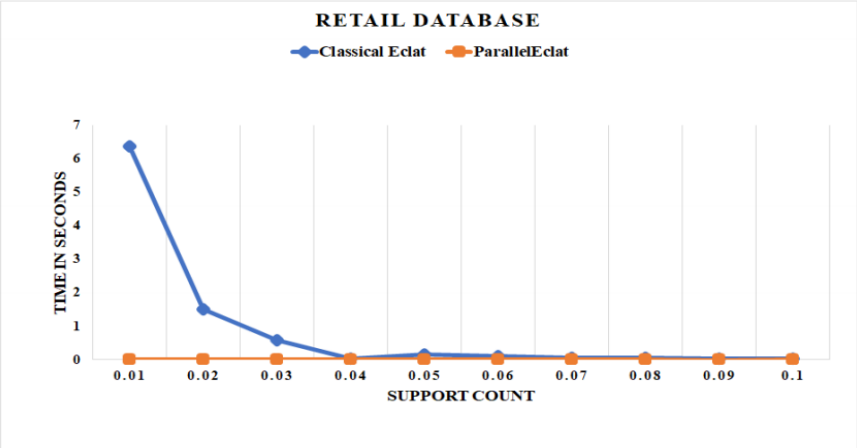

Figure 4.4: *Time Analysis for Retail Dataset*

Fig. 4.5 and Fig. 4.6 demonstrates the performance result with respect to the minimum support count ratio on Nursery dataset. The nursery dataset is a real database that has 12960 transactions in the transaction database and has average transaction size of 48. It can be seen clearly from the graph that the Parallel Eclat Algorithm runs much faster than the Classical Eclat as the support count ratio decreases.

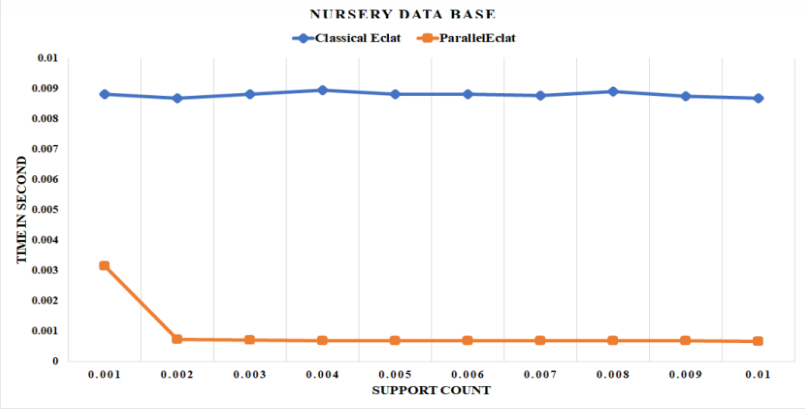

Figure 4.5: *Time Analysis for Nursery Database*

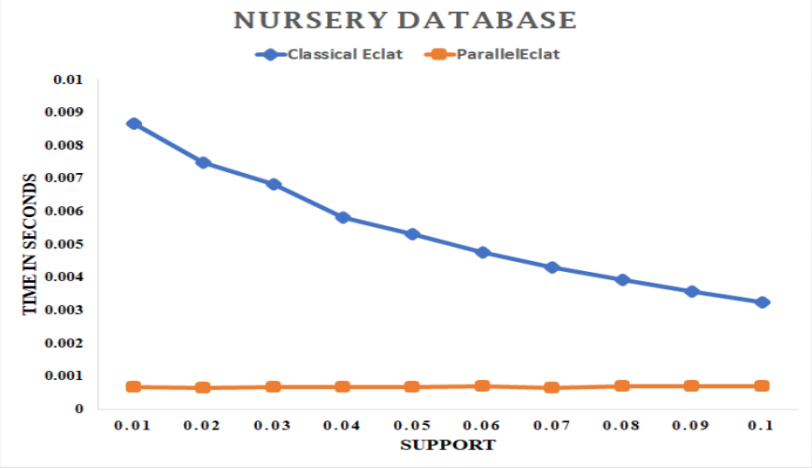

Figure 4.6: *Time Analysis for Nursery Dataset*

Fig. 4.7 and Fig. 4.8 demonstrate the performance result with respect to minimum support count ratio on US Census dataset. US Census dataset is a real database that has 2458285 transactions. As shown in figure, Parallel Eclat Algorithm runs faster than Classical Eclat Algorithm for large datasets for lower values of minimum support count and higher values of minimum support count.

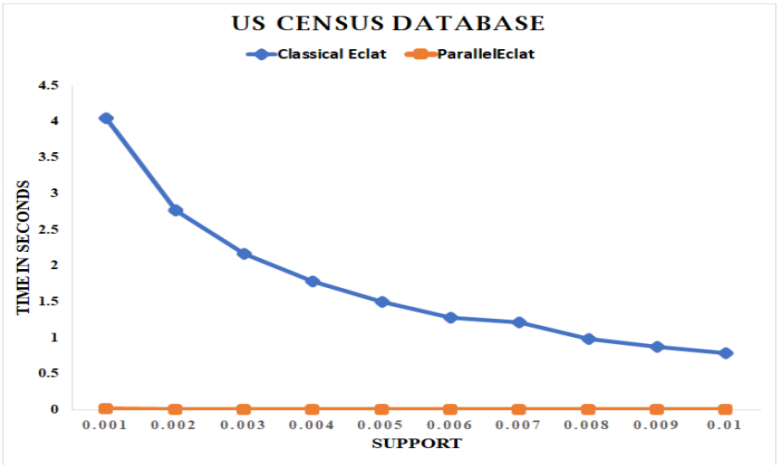

Figure 4.7: *Time Analysis for US Census Dataset*

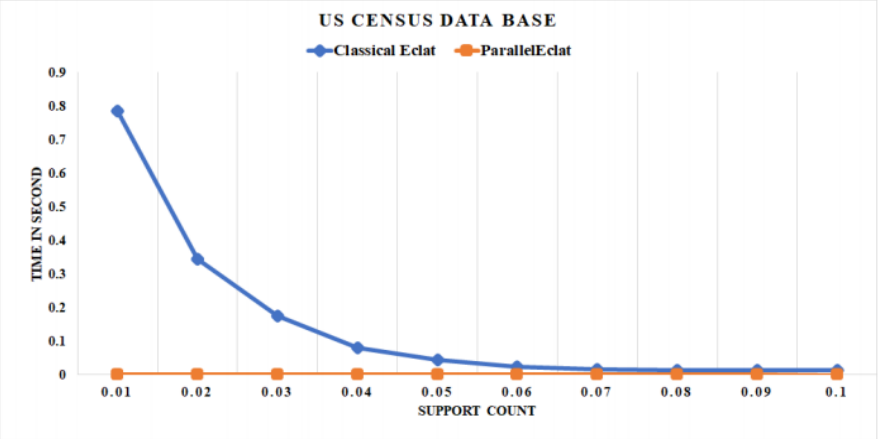

Figure 4.8: *Time Analysis for US Census Dataset*

Fig. 4.9 and Fig.4.10 demonstrate the performance of Parallel Eclat Algorithm on the US Income dataset. US Income dataset is a real database with 299285 transactions in the transaction database. Fig. 4.9 and Fig. 4.10 show that the performance of Parallel Eclat Algorithm is better than Classical Eclat Algorithm.

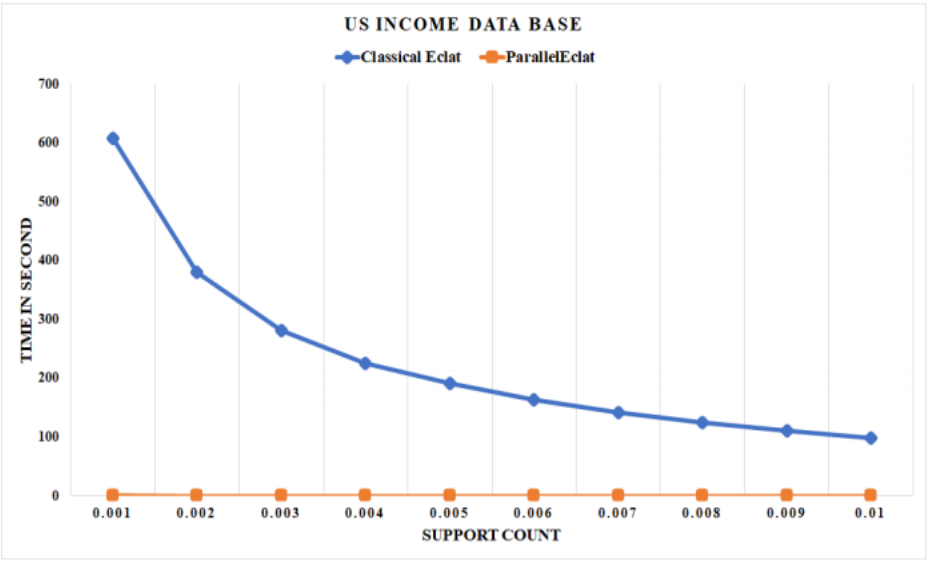

Figure 4.9: *Time Analysis for US Income Dataset*

Thus, it can be observed from the graphs that the proposed algorithm has better performance than the Classical Eclat algorithm for lower values of support count, larger average transaction size and large size of the transaction database.

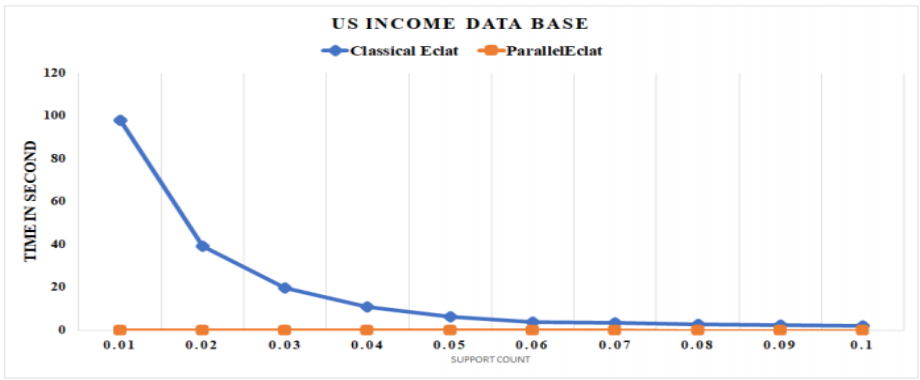

Figure 4.9: *Time Analysis for US Income Dataset*

V. Conclusion

In this paper, we described and implemented a new approach for implementing classical Eclat Algorithm in parallel using optimized candidate item- set representation. It has been observed from the experiments and results that the Parallel Eclat algorithm in parallel has better performance than the classical Eclat algorithm in terms of time. The Performance of the Parallel Eclat algorithm is better even for the lower minimum support count value, large transaction database and large average transaction size.

VI. Future Work

In the future the Parallel Eclat algorithm can be implemented for Multicore GPU that can improve theperformance of the algorithm further.

References

[1]. K. Jiawei, Han Micheline, *Data Mining Concepts and Techniques*.

[2]. R. Agrawal, T. Imieliski, and A. Swami, Mining association rules between sets of items in large databases,ACM SIGMOD Rec.,22(2), 1993, 207-216.

[3]. S. Dutt, N. Choudhary, and D. Singh, An Improved Apriori Algorithm based on Matrix Data Structure, *Global Journal of Computer Science and Technology: C Software & Data Engineering*,14(5), 2014.

[4]. U. Grag, and M. Kaur, ECLAT Algorithm for Frequent Itemsets Generation, *International Journal of Computer Systems*,1(3), 2014,1-4.

[5]. NVIDIA Corporation, "Datasheet: NVIDIA Kepler Next-Generation Cuda Compute Architecture," Nvidia White Pap., 2012

[6]. C. Silvestri, and S. Orlando, gpuDCI: Exploiting GPUs in frequent itemset mining, *Proc. - 20th EuromicroInt. Conf. Parallel, Distributed and Network-Based Processing* PDP,2012, 416–425.

[7]. F. Zhang, Y. Zhang, and J. Bakos, GPApriori: GPU-accelerated frequent itemset mining, *Proc. - IEEE Int. Conf. Cluster Computing ICCC*, 2011, 590-594.

[8]. R. Agarwal and J. Shafer, Parallel Mining of Association Rules(IBM Almaden Research Center).

[9]. T. Li, and D. Luo, A New Improved Apriori Algorithm Based on Compression Matrix,*Springer International Publishing Switzerland*, 2014, 1-15.

[10]. D. Bhalodiya, K. M. Patel, and C. Patel, An Efficient Way to Find Frequent Pattern with Dynamic Programming Approach, 2013 Nirma University*Int. Conf. on Engineering*.,2013, 1-5.

[11]. L. Liu, E. Li, Y. Zhang, and Z. Tang, Optimization of Frequent Itemset Mining on Multiple-Core Processor, *Int. Conf. Very Large Data Bases*,2007, 1275–1285.

[12]. X. Zeng, J. Lv, J. Li, and W. Luo, An Optimized Apriori Algorithm Based on Sparse Matrix for Intrusion Detection,*The Open Cybernetics & Systemics Journa**l***, 8, 2014, 8-11.

[13]. W. Fang, M. Lu, X. Xiao, B. He, and Q. Luo, Frequent itemset mining on graphics processors, *Fifth International Workshop on Data Management on New Hardware*.,2009, 34–42.

Janki Kansagra. " Mining Frequent Patterns with Optimized Candidate Representation on GPU using Parallel Eclat Algorithm" IOSR Journal of Computer Engineering (IOSR-JCE) 21.1 (2019): 16-25.