Journal of Theoretical and Applied Information Technology
31st July 2021. Vol.99. No 14

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

P-BBA: A MASTER/SLAVE PARALLEL BINARY-BASED ALGORITHM FOR MINING FREQUENT ITEMSETS IN BIG DATA

ALIYA NAJIHA AMIR[1] , HITHAM SEDDIG ALHASSAN ALHUSSIAN[2] , SALLAM OSMAN FAGEERI[3] , ROHIZA AHMAD[4]

[1,2,4]Department of Computer and Information Sciences, Universiti Teknologi Petronas, Malaysia

[3]Department of Information Systems, University of Nizwa, Sultanate of Oman

E-mail: [1]aliya_19000192@utp.edu.my, [2]seddig.alhussian@utp.edu.my, [3]sallam@unizwa.edu.om, [2]rohiza_ahmad@utp.edu.my

ABSTRACT

Frequent itemset mining is a data mining technique to discover the frequent patterns from a collection of databases. However, it becomes a computational expensive task when it is used for mining large volume of data. Hence, there is a necessity for a scalable algorithm that can handle bigger datasets. Binary-based Technique Algorithm (BBT) can simplify the process of generating frequent patterns by using bit wise operations and binary database representation. However, it still suffers with the problem of low performance when dealing with high volume of data and a minimum values of support threshold to generate the list of frequent itemset patterns. This is due to its design which run in a single thread of execution. This research proposed a Parallel Binary-Based Algorithm (P-BBA) to solve the mentioned problem. The proposed algorithm is designed with collaborative threads which simultaneously work together to generate frequent itemsets in a big data environment. A master/slave architecture is used to fit the algorithm with distributed computing platform. The obtained results showed significant reductions in execution time when using the proposed parallel binary-based algorithm.

Keywords: *Big Data mining, Distributed Framework, Frequent Itemsets Mining, Parallel Frequent Item Mining*

1. INTRODUCTION

Data mining is a powerful emerging technology to discover hidden knowledge that can be in a form of pattern, correlations, relationships, and anomalies [1]. When going into the context of data mining, Frequent itemset mining (FIM) is considered as the core for association rule mining [2]. FIM is used to find which particular set of items may contain high number of occurrences from a list of database transactions [3]. Information of frequent items extracted from database is crucial to discover the hidden patterns between data and relevant association rules. The occurrence of itemsets provide a valuable information to support critical decision and prediction.

However, frequent itemsets mining is a computationally expensive task due to its algorithm which turn an initially subset problem into an exponential growth of complexity. Even though that FIM algorithm may cause a huge computation cost, its usefulness in discovering hidden information lets FIM into getting tremendous popularity in the industry.

The original motivation to generate frequent items is from the application of supermarket transactional database [4]. FIM is used by the supermarket company to find the items that are frequently by together with other items in a single customer's transaction. Generating suggestion and recommendation helps in planning the business strategies and leads to gain more profit. Other fields that benefit from the application of FIM algorithm include bioinformatics and health environments [5]. FIM is widely used in DNA analysis by detecting the behavior of chromosomal mutation in DNA cells [6]. The uses of FIM helps in diagnosing the breast cancer detection at the early stage, thus increase the survival rate of cancer patients. Another application of frequent itemsets mining is

to improve text summarization of biomedical paper. Researchers in biomedical area face difficulties to access the biomedical bibliographic text since it is consisting of huge databases with more than 25 million references [7]. Various of different applications of FIM have proven that FIM plays an important role especially in the field of business analytics and bioinformatics.

Among of the well-known existing state-of-the-art FIM algorithm that have been frequently referred by many researchers are Apriori, FP-Growth and Eclat [8-10]. The algorithm has been used and extended by other authors [2,4,11,12]. Even though that these algorithms have been widely used and applied, those algorithms still have several drawbacks especially in terms of the time performance and the usage of space [13]. Both of these issues can be solved using the Binary-based Technique Algorithm (BBT) [13]. This algorithm make use the benefit of bit wise operation for simplifying the process of generation frequent itemsets [13,14].

Even though that the BBT algorithm is proven to have a better performance than the existing state-of-the-art algorithm, when it is applied in large scale of data, it still suffers with the problem of slow execution time. Applications that use the FIM algorithm typically consists of large volume of data and may fail to generate any output due to the explosion of the generated frequent itemsets. Since it is dealing with massive data, the existing FIM algorithm like Apriori, FP-growth and Eclat deal with many difficulties [15].

The usefulness and benefits offered by FIM give the motivations to address the challenge of mining frequent itemsets in big data. Therefore, there is an important need to implement a further improved FIM algorithm that may address the issues of processing large databases. Researchers are looking forward towards improved FIM algorithms with high performance as a better alternative [16-19].

The aim of this research is to further enhance the performance of Binary-based Technique Algorithm (BBT) in terms of execution time when it is applied in big data environment by using a parallel approach. Therefore, we formulate a new Parallel Binary-based Algorithm (P-BBA) for mining frequent itemsets in big data that generate the results within an acceptable time frame. This algorithm is designed with a master/slave thread model to fit with distributed computing platform like Apache Sparks framework.

The new proposed method allows the master thread to monitor and control the execution of the slave threads in order to generate frequent itemsets. It also has the capability to control the quantity of working threads to adapts with the environment of datasets in terms of capacity, density, complexity and the availability of the underlying computing resources. Well-known datasets that are used for frequent itemset mining problem will be used to measure the performance in term of execution time.

2. PROBLEM STATEMENT

When considering big datasets, well-known existing frequent itemset mining (FIM) algorithm suffers with the problem of slow execution and may even fail to produce any output due to the explosion of generated frequent itemsets. The algorithm initially deals with a subset problem by default which turn into an exponential growth of complexity. The Binary-based Techniques Algorithm (BBT) make use the benefits of binary representation to simplify the process of identifying frequent pattern, thus reduce the total execution time as well as the memory consumption (Fageeri et al., 2014). However, when considering big datasets, it still suffers with the problem of low performance in terms of execution times. This is cause by its design that run in a single execution thread. With the mentioned problem, there is a need to further improve the Binary-based Technique Algorithm (BBT) to adapt with the architecture of distributed computing platform. For distributed computing platform, Apache Spark framework will be chosen instead of Map/Reduce framework due to its benefit of using in-memory processing and its capability to generate real-time analytics for big data.

3. RESEARCH OBJECTIVES

• To formulate a Master/Slave Parallel Binary-based Algorithm for mining frequent itemset in big data environment.

• To design and implement the proposed algorithm in a Master/Slave architecture in order to adapt with a distributed computing framework.

• To validate the proposed algorithm using well-known big datasets and conduct CPU profiling simulations and evaluate the performance in terms of total execution time.

4. LITERATURE REVIEW

Frequent itemset mining (FIM) provide a valuable information to support critical decision and prediction. R Agrawal et al. [8] proposed an Apriori algorithm that use prune techniques and different candidate generation method to generate frequent itemsets. This method is able to reduce the number of candidate itemsets. It uses the level-wise search to explore (k + 1) itemsets to mine itemsets that have a high amount of frequency. However, the methods have complex generation of candidate itemsets and require a repetitive scanning throughout each different iteration.

Han et al. [10] proposed an improved Apriori algorithm called FP-growth Algorithm. It solves 2 bottle neck arise in Apriori algorithm. FP-Growth can be adapted in larger volume of database by generating the list of frequent items by creating its own data structure which is the FP-Tree from using a prune technique for infrequent items. FP-Growth is able to reduce the total execution time to process the output results but it needs the uses of more memory consumption. This approach is also more complicated and difficult to be implemented compared to the other approach.

Zaki [9] proposed Eclat or also knows as Equivalence Class Transformation Algorithm which use the depth-first search strategy that create a vertical data format. The calculation of number of supports are much more efficient when the database transaction is represented in a vertical representation. What makes Eclat differ with Apriori algorithm is that the generation (k+1) itemsets is not require. Therefore, multiple scanning and unnecessary consumption of time and memory can be avoided. All items are saved in a list (TID list) while intersection-based approach is used for computation itemset's support [20]. Every item inside the TID list have its own unique transaction identifiers (TIDs).

Fageeri and Ahmad [13] proposed a new FIM algorithm called The Binary-based Techniques Algorithm (BBT). This new algorithm is proven to have a better performance than the previous three algorithms (Apriori, FP-Growth and Eclat). The authors highlighted the drawbacks of those previous algorithm especially in terms of their execution time performances and the uses of inefficient approach in the process of comparing the items. Those method scans every possible combination and calculate the occurrences of all those combinations. The requirement of multiple scanning leads to unnecessary consumption of time. The new BBT algorithm is reported to be able to tackle those issues. It uses the advantage of binary implementation to compare the frequency of items efficiently. Other algorithm requires a multiple scanning while BBT algorithm only scan the original database in a single cycle. The BBT algorithm is also not required to go through the unnecessary needs for sorting the datasets. BBT algorithm further improves the existing algorithm and remove issues that are arises in previous algorithms such as delays of execution time and high consumptions of memory. Furthermore, it also has another additional ability for decision support by scanning and generating the infrequent itemsets.

Aggarwal and Han [15] discussed about the challenges and difficulties that are faced by FIM algorithm like Apriori, FP-growth and Eclat. These algorithms suffer with many difficulties due to the challenge of adapting with high volume of data. The current volume of data is increasing with the growing industries and increasing of internet user [21]. The massive amount of data also leads to the explosion of generated frequent itemsets. The initial Apriori algorithm consume a very long execution time when it is deal with high-scale candidate itemsets since candidate (k + 1) itemsets are constructed through the self-join of frequent k-itemsets (R Agrawal et al., 1996; Yuan, 2017). High number of candidates itemsets also leads to exponential growth of those algorithms. Multiple scanning is needed for every iteration to generate the frequent itemsets [15, 22]. Therefore, researchers are looking forward towards improved FIM algorithms with high performance as a better alternative [16-19]. Executing FIM algorithms by using parallel approach with the implementation of multiple cluster nodes would address the time complexity problem [3].

Hazarika and Rahman [23] proposed a parallelize Eclat algorithm called MR_Eclat algorithm to further improve the performance of Eclat algorithm. The proposed method is scalable to large data set with less cost and better performance. The input data are divided among different nodes. MR_Eclat scan the database and transform to vertical database. The database will then go through the synchronous phase, asynchronous phase and reduction phase. Implementation of Hadoop helps in managing any failure of nodes in the cluster. MR_Eclat has proven to have a better performance than the non-parallel Eclat algorithm. However, MR_Eclat algorithm does not give any significant impact when working with small data sets. It might

give similar run time or worse 27 than the non-Hadoop system for the same algorithm.

Li et al. [24] proposed an improved FP-growth algorithm called Parallel FP-Growth (P-FP) algorithm by using the method of distributing the mining task into different partition of independent parallel tasks. The parallel FP-Growth algorithm is proven to have a better performance than the non-parallel FP-growth algorithm. The uses of independent FP-tree based on the FP-growth algorithm gives a significant improvement in terms of runtime and memory which gives the benefit to exploits the limitations of GPU memory. However, if the numbers of minimum support threshold is too low, P-FP algorithm could not handle the program. It also generates a long itemset for each sub transactions. The authors have also suggested to use the P-FP algorithm together with the implementation of distributed framework such as Hadoop MapReduce framework and Apache Spark framework for much better performance. The parallelism of most FIM algorithms can be efficiently achieved by using such framework [25].

Kourtesis et al. [26] proposed to execute Machine Learning algorithms on multicore architectures used the implementation of Hadoop MapReduce framework. It is proven that Hadoop MapReduce framework has the ability to automatically handling failure. The complexity of fault-tolerance can be efficiently handled using this feature. It simplifies the development of application in distributed environment and better parallel performance can be obtained as this framework provides a parallel design pattern. However, recent studies reveals that it is not efficient enough to implement MapReduce framework in parallelizing Apriori-based algorithms [19].

Qiu et al. [27] proposed a further Apriori algorithm called YAFIM (Yet Another Frequent Itemsets Mining) algorithm to to parallelize the Apriori algorithm using Apache Spark framework. This algorithm is an extended version of Apriori algorithm based on Apache Spark framework. This framework is chosen to cope with the overhead issue caused by the launch of new MapReduce jobs. The motivation of this approach is by the use of its in-memory-based data process, an iterative computing framework, compared to MapReduce which use a disk-based to process the dataset [28]. The process of data using memory is much faster than using disk and it also reduce the memory usage [29]. Therefore, Spark's performances outperform Hadoop's performance especially when dealing with iterative computations. Spark framework could achieve up to 18× computation speedup in average for various benchmarks [30].

The best performance of FIM algorithm could be achieve by using parallelization together with the implementation of Apache Spark Framework. With these findings, there is a need to implement the stated method for BBT algorithm as BBT algorithm is proved to have the best performance compared to the previous FIM algorithm (Apriori, FP-Growth and Eclat).

5. METHODOLOGY

The implementation of Frequent itemset mining (FIM) in data mining is crucial to discover the hidden patterns between data and relevant association rules to support critical decision and prediction. Among of the popular algorithms that have been widely use and applied are Apriori, FP-Growth and Eclat. However, Binary-based Technique algorithm (BBT) are proven to have a better performance than the previous algorithm through the implementation of binary and bit wise operations. Even though that the BBT algorithm is able to solve the issues arise in the previous algorithm, it stills suffer with the problem of slow execution time when dealing with high volume of data. The usefulness and benefits offered by FIM algorithm give the motivations to address the challenge of mining frequent itemsets in big data.

The main factor that leads to this problem is because of the design of the previous FIM algorithm that only run in a single thread of execution. It is proven that application which consist with high volume of data can be effectively deal with parallel approach [23,24]. Executing FIM algorithms by using parallel approach with the implementation of multiple cluster nodes would address the time complexity problem [3]. This alternative has also leads to explosion of parallel computing platforms such as Hadoop MapReduce framework and Apache Spark framework. Among of many available type of framework, recent studies have found that an attempt to parallelize Apriori algorithms based on the Spark frameworks could achieve up to 18× computation speedup in average for various benchmarks compared to Hadoop framework [30].

In this proposed model, the Binary-based Technique Algorithm (BBT) is chosen to address the challenge of mining frequent itemsets in big data context as the performance of BBT algorithm outperform the previous FIM algorithm due to the use of binary and bit wise operations. The main

objective is to solve the time complexity problem cause by huge load of data to be process. Executing FIM algorithms by using parallel approach with the implementation of multiple cluster nodes would address the time complexity problem. Therefore, a parallel approach is used to improve the performance of BBT algorithm. The formulation of this new proposed algorithm can be further improved with the implementation of parallel computing framework such as Spark framework and Hadoop framework. Apache Spark framework is chosen as it could achieve up to 18× computation speedup than Hadoop framework. Therefore, this new proposed algorithm is design in a master/slave model to fit in the architecture of Apache Spark framework. In conclusion, we proposed a new parallel Binary-based Technique algorithm implemented with master/slave architecture for mining frequent itemsets in big data.

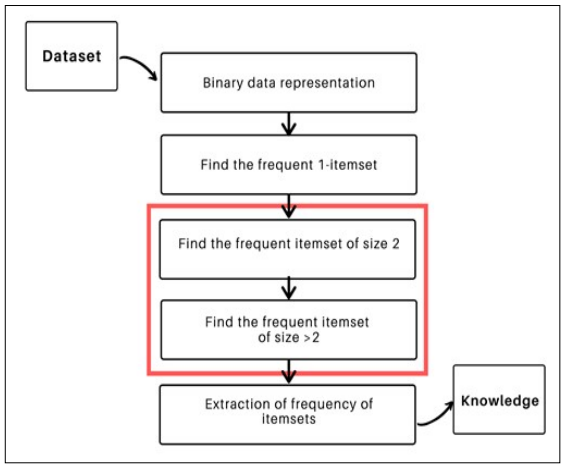


Figure 1: Contribution area from the execution flow

The figure shows the execution flow and the phases involve in Binary-based Technique algorithm (BBT) to generate the frequent itemsets. The process begins with the read of data from the dataset and generate a binary representation. Next, the frequency of all items is calculated and any items that passed the minimum supports threshold is stored. Then the system finds the frequency of itemset of size 2 and size larger than 2. Finally, the frequency of itemsets is extracted to form a knowledge.

The aim of this research study is to improve the performance of Binary-based Technique algorithm (BBT) in terms of execution time. Based on the figure, the highlighted part took a long time to execute due to the challenges of processing a huge amount of data which cause a time complexity problem. When finding the frequency of itemset size 2 and size larger than 2, a long process is required to compare the occurrence of items through each candidate itemsets. Therefore, the role of this proposed algorithm is to use the parallel approach by decomposing this part into sub-problem that can be executed in parallel. The parallel approach and master/slave architecture based on Apache Spark framework is selected to improve the performance of the BBT algorithm in terms of execution time.

Input: Database transaction, DB;
Itemset, SortedSet; Band List of BitSet, Binary dataset;
Output: Association rule; Hashtable, freqHT

1. Start
2. Call getItemSet()
3. Call buildBinaryDB()
4. Call generateFrequentItemSets()
6. Call createMultipleThread()
7. Call getFrequencyItemSet()
8. Call printResults()
9. End

Figure 2: Pseudocode of the Master Parallel Binary-based algorithm.

Variable	Definition
DB	Database transaction
freqHT	A Hash table that stores frequent itemsets
BitSetDB	Binary representation of database
minSup	The value of minimum support threshold
threads	Number of threads to be assigned
BinDB	Database in binary representation
mf1stpass	List of items that pass the minimum support threshold
freq	The value of frequencies of items
frequentItems	List of frequent items of size 1
frequentItems_n	List of frequent items of size > 1
frequentItemSet	Itemsets that pass the minimum support threshold
DBSize	Total number of lines contain inside the database

Table 1: Definition of the variables.

The master algorithm starts by preparing the list of itemsets by calling the *getItemSet* procedure (line 2) and then proceeds to create the binary representation of the database by calling *buildBinaryDB* procedure (line 3). The result of these two procedures is demonstrated by Fig. 3 below.

Journal of Theoretical and Applied Information Technology
31st July 2021. Vol.99. No 14

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

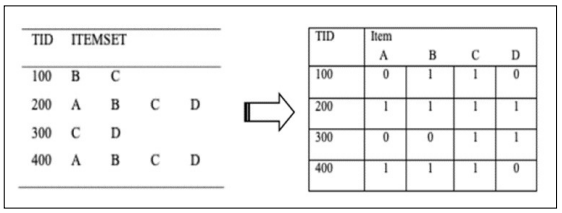


Figure 3: Generation of database binary representation [13].

Any required calculation to find the frequency of itemsets will only be based in binary. All items in the database are consist of digit 1 and 0 that represent the occurrence of the corresponding items.

Before creating the parallel threads and execute them, the master needs to calculate the list of frequent itemsets of size 1. This is done by calling the procedure *generateFrequentItemSets* (line 4). The result of executing this procedure is demonstrated by Fig. 4.

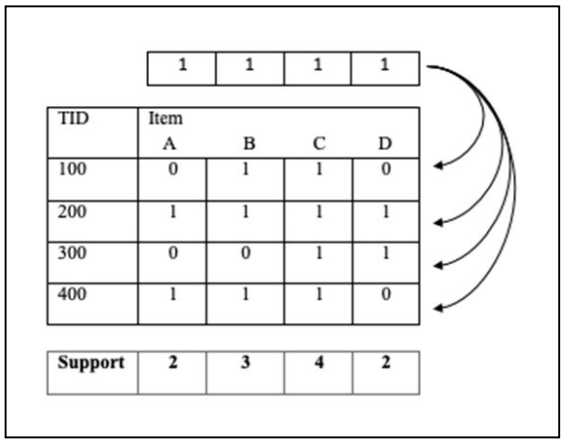


Figure 4: Process of calculation the frequency of each item [13].

Once the list of frequent itemsets of size 1 has been calculated, the master will create multiple threads and divide the binary database into equal portions. The pseudo code of the procedure is listed in Fig. 5 as shown below. The above procedure is done by calling the procedure *createMultipleThread* (line 6). The total numbers of lines of dataset will be divided by the number of specified threads. With this method, each thread will process an equal line of database.

Input: Database size, DBSize;
Threads assigned, NoOfThreads;
Output: timeStamp; NoOfFrequentItemsets; NoOfAssociationRules;

```
1.  Start
2.      Get number of threads
3.      Get database size
4.      Set step = DBSize/NoOfThreads
5.      Set startRow =0, endRow = step
6.      For each thread, binaryThreads[index]
7.          binaryThreads[index].initialize(startRow, endRow)
8.          binaryThreads[index].start()
9.          startRow = endRow, endrow = endrow + step
10.     end for
11.     binaryThreads[index].join
12.     print timeStamp
13.     print number of frequentItemsets
14.     print number of associationRules
15. End
```

Figure 5: Pseudocode of the parallel threads.

Once all threads have been assigned their respective portions, each thread will start calculating the list of frequent itemsets of size > 1 based on its designated portion. Fig. 6 below demonstrate how each thread calculates the list of itemsets of size > 1.

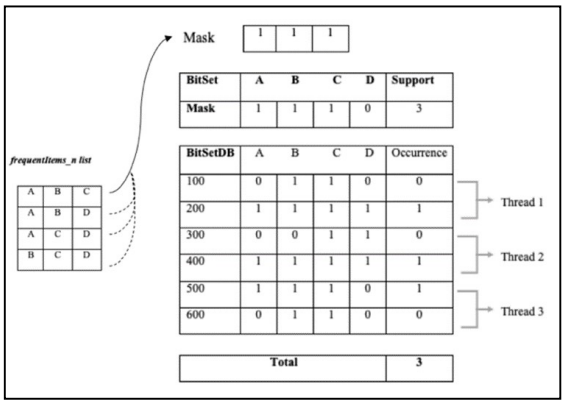


Figure 6: Frequency analysis of frequent itemset size > 1.

Each thread compares the same masked *frequentItemSet_n* with the respective lines of binary database. All assigned threads are executed simultaneously and then integrate the results after all threads finish its task. Once the result is generated, the next step is to compare the number of support and store only the element that have the frequency higher than the minimum support into the *frequentItemSets_n* list. The same procedure and method are repeated for the next set of itemsets in the *frequentItems_n* list. Itemsets that consist of a

high value of frequency provides a valuable information. The next phase is generating and displaying the output results through the *printResults* procedure which print the total execution time and extract the association rule.

6. EXPERIMENTAL RESULTS

This section presents the experimental result based on the proposed Parallel Binary-based algorithm (P-BBA). The objective of P-BBA algorithm is to improve the performance of Binary-based Technique Algorithm (BBT) by implementing multiple execution thread. This experiment evaluates the performance and observe the comparison in terms of total execution time. In this context, the meaning of total execution time is the time taken for the program to complete the execution and generate output results.

The result of this experiment is evaluated using a set of well-known datasets for frequent itemset mining problem provided by FIMI repository. Results of execution time is generated for T10I4D100K datasets, one of the popular datasets used to evaluate the performance of mining frequent itemsets. It is generated by IBM Market-Basket Synthetic Data Generator.

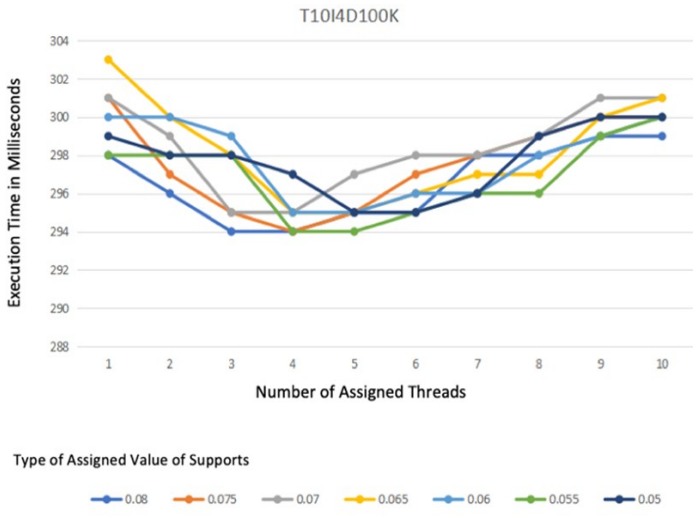
Results of total execution time is generated by using 2 different variable which are the number of threads and the range of support.

Range of support varies started with the range of 0.08 until 0.05. For each range of support, the total execution time is generated with different numbers of threads started with the uses of 1 thread until the uses of 10 threads. This variable is taken to observe and compare on how different number of threads may impact the total execution time. The result is demonstrated in the Table 2 as shown below.

Table 2: Total execution time based on dataset T10I4D100K.

	Number of Threads									
Support	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0.08	298	296	294	294	295	295	298	298	299	299
0.075	301	297	295	294	295	297	298	299	300	300
0.07	301	299	295	295	297	298	298	299	301	301
0.065	303	300	298	295	295	296	297	297	300	301
0.06	300	300	299	295	295	296	296	298	299	300
0.055	298	298	298	294	294	295	296	296	299	300
0.05	299	298	298	297	295	295	296	299	300	300

Graph 1 presents the result of total execution time of the proposed Parallel Binary-based Algorithm (P-BBA) when it is executed across different range of support with different numbers of threads. Total execution time refers to the calculated amount of time for the program to be executed completely and produce an output result of frequent itemsets. Based on the above graph 1, it clearly shows that the algorithm achieves its best performance when 4-5 numbers of thread is used during the execution. For each type of support, the slowest execution time occurs either when it is using only 1 or 10 number of threads, which is the



Graph 1: The results of total execution time of P-BBA algorithm based on dataset T10I4D100K.

highest numbers of threads. Using only 1 number of threads will always resulted in slower execution time than using 2 number of threads. When the numbers of threads increase, the total execution time will become much faster. The load of work have been shared together equally by multiple threads. This experiment has proved that the total execution time for generating the frequent itemsets become faster when using multiple execution thread rather than running it in only 1 whole single execution.

However, for each type of range of support, there will always be a certain point where the graph become constant or started to gradually increase. This is due to the limitation of available resources numbers of threads. The programs reach its optimum utilization when it is executed using 4 or 5 numbers of threads. In conclusion, the program could not achieve its optimum performances if the numbers of threads used is larger than the available numbers of thread resources. This situation is cause by the queuing and waiting for a non-available thread that resulted a delay during the program execution. The results from the above graph 1 also provide the good optimum number of threads to be implemented when executing the program.

7. EFFICACY OF THE PROPOSED ALGORITHM

The performance of BBT algorithm is able to outperform the other previous FIM algorithm through the use of its binary and bit wise operation. However, it still suffers with the time complexity problem when processing huge amount of data. The main factor of this problem is due to its design which only run in a single thread of execution. The parallel approach and multiple execution are chosen to improve the performance of BBT algorithms. The proposed parallel approach also uses the master/slave architecture to fit in with Apache Spark framework.

The results of applying multiple cluster node have shown an improvement in the total execution time to generate the frequent itemsets.

The novelty of the current works lies in the fact that the result from the uses of parallel approach and master/slave architecture together with the implementation of bit wise operation for mining frequent items has been proven.

With this improvement of time performance, the proposed algorithm can be applied to generate frequent itemsets in big data environment.

8. CONCLUSIONS AND FUTURE WORK

This paper presented a proposed Parallel Binary-based Algorithm (P-BBA), an improved frequent itemset mining algorithm in terms of execution time based on the Binary-based Technique Algorithm (BBT). The demonstrated experimental results proves that uses of multiple thread in the process of generating frequent itemsets can shorten the total execution time compared to using only single thread of execution. The master/slave architecture helps in coordinating numbers of threads to work together collaboratively in order to reduce the processing time. Compared to the previous existing frequent itemset mining algorithms, this new solution can be declared as another better alternative to counter the issues in processing a massive amount of data in the big data application. Therefore, the application of this alternative method is able to adapt with the current revolution of big data as most of business industries are moving towards digitalization of data.

The proposed method has proven to have a better performance in terms of execution time. However, there is still some potential area to be discovered to further enhance the performance of the proposed algorithm. Among of the future work is the implementation P-BBA algorithm with GPUs. Even though that the implementation of parallel execution has improved the performance, GPUs can be implemented to further improve the performance in terms of the execution time. It also uses the master/slave model that divide the entire database into partitions and splits the partition into numbers of GPU worker. GPU provides a higher number core compared to CPU. Therefore, more available numbers of threads can be implemented for the program. The higher the number of threads and parallel execution to be implemented, the faster the execution time to generate output.

9. ACKNOWLEDGEMENT

This research is being supported by the university, UTP through the Yayasan Universiti Teknologi Petronas (YUTP) grant: 015LC0-286.

REFRENCES:

[1] Yan H, Yang N, Peng Y, Ren Y. Data mining in the construction industry: Present status, opportunities, and future trends. Automation in Construction. 2020 Nov 1;119:103331.
[2] Gang WX. A Summary of Research on Frequent Itemsets Mining Technology. Procedia Computer Science. 2018 May 1;131(C):841-6.
[3] Djenouri Y, Djenouri D, Belhadi A, Cano A. Exploiting GPU and cluster parallelism in single scan frequent itemset mining. Information Sciences. 2019 Sep 1;496:363-77.
[4] Pramod S, Vyas OP. Survey on frequent item set mining algorithms. International journal of computer applications. 2010;1(15):86-91.
[5] Zou Q, Li XB, Jiang WR, Lin ZY, Li GL, Chen K. Survey of MapReduce frame operation in bioinformatics. Briefings in bioinformatics. 2014 Jul 1;15(4):637-47.
[6] Sinha A, Sahoo B, Rautaray SS, Pandey M. Improved framework for breast cancer prediction using frequent itemsets mining for attributes filtering. In2019 International Conference on Intelligent Computing and Control Systems (ICCS) 2019 May 15 (pp. 979-982). IEEE.
[7] Rouane O, Belhadef H, Bouakkaz M. Combine clustering and frequent itemsets mining to enhance biomedical text summarization. Expert Systems with Applications. 2019 Nov 30;135:362-73.
[8] Agrawal R, Mannila H, Srikant R, Toivonen H, Verkamo AI. Fast discovery of association rules. Advances in knowledge discovery and data mining. 1996 Feb 1;12(1):307-28.
[9] Zaki MJ. Scalable algorithms for association mining. IEEE transactions on knowledge and data engineering. 2000 May;12(3):372-90.
[10] Han J, Pei J, Yin Y, Mao R. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data mining and knowledge discovery. 2004 Jan;8(1):53-87.
[11] Agapito G, Guzzi PH, Cannataro M. Parallel and distributed association rule mining in life science: A novel parallel algorithm to mine genomics data. Information Sciences. 2018 Jul 26.
[12] Han J, Haihong E, Le G, Du J. Survey on NoSQL database. In2011 6th international conference on pervasive computing and applications 2011 Oct 26 (pp. 363-366). IEEE.
[13] Fageeri SO, Ahmad R, Baharudin BB. Bbt: An efficient association rules mining algorithm using binary-based technique. International Journal of Advancements in Computing Technology. 2014 Jul 1;6(4):14.
[14] Fageeri SO, Ahmad R. An efficient log file analysis algorithm using binary-based data structure. Procedia-Social and Behavioral Sciences. 2014 May 15;129:518-26.
[15] Aggarwal CC, Bhuiyan MA, Al Hasan M. Frequent pattern mining algorithms: A survey. InFrequent pattern mining 2014 (pp. 19-64). Springer, Cham.
[16] Suneel CV, Prasanna K, Kumar MR. Frequent data partitioning using parallel mining item sets and MapReduce. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 2017;2(4).
[17] Shirke D, Varshney D. Parallel Mining of Frequent Itemsets in Hadoop Cluster Having Heterogeneous Nodes. International Journal. 2017 Jul;5(7).
[18] Xun Y, Zhang J, Qin X. Fidoop: Parallel mining of frequent itemsets using mapreduce. IEEE transactions on Systems, Man, and Cybernetics: systems. 2015 Jun 15;46(3):313-25.
[19] Bharathi T, Krishnakumari P. A comparative analysis on efficiency of contemporary association rule mining algorithm. In2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS) 2016 Jan 22 (Vol. 1, pp. 1-9). IEEE.
[20] Robu V, dos Santos VD. Mining frequent patterns in data using apriori and eclat: A comparison of the algorithm performance and association rule generation. In2019 6th International Conference on Systems and Informatics (ICSAI) 2019 Nov 2 (pp. 1478-1481). IEEE.
[21] Khan N, Alsaqer M, Shah H, Badsha G, Abbasi AA, Salehian S. The 10 Vs, issues and challenges of big data. InProceedings of the 2018 international conference on big data and education 2018 Mar 9 (pp. 52-56).
[22] Yuan X. An improved Apriori algorithm for mining association rules. InAIP conference proceedings 2017 Mar 13 (Vol. 1820, No. 1, p. 080005). AIP Publishing LLC.
[23] Hazarika M, Rahman M. Mapreduce based eclat algorithm for association rule mining in datamining: Mr _ Eclat. International Journal of Computer Science and Engineering. 2014 Jan;3(1):19-28.

[24] Li H, Wang Y, Zhang D, Zhang M, Chang EY. Pfp: parallel fp-growth for query recommendation. InProceedings of the 2008 ACM conference on Recommender systems 2008 Oct 23 (pp. 107-114).
[25] Salah S, Akbarinia R, Masseglia F. Data placement in massively distributed environments for fast parallel mining of frequent itemsets. Knowledge and Information Systems. 2017 Oct;53(1):207-37.
[26] Kourtesis D, Alvarez-Rodríguez JM, Paraskakis I. Semantic-based QoS management in cloud systems: Current status and future challenges. Future Generation Computer Systems. 2014 Mar 1;32:307-23.
[27] Qiu H, Gu R, Yuan C, Huang Y. Yafim: a parallel frequent itemset mining algorithm with spark. In2014 IEEE International Parallel & Distributed Processing Symposium Workshops 2014 May 19 (pp. 1664-1671). IEEE.
[28] Gassama AD, Camara F, Ndiaye S. S-FPG: A parallel version of FP-Growth algorithm under Apache Spark™. In2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA) 2017 Apr 28 (pp. 98-101). IEEE.
[29] Shoro AG, Soomro TR. Big data analysis: Apache spark perspective. Global Journal of Computer Science and Technology. 2015 Feb 21.
[30] Mavridis I, Karatza H. Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark. Journal of Systems and Software. 2017 Mar 1;125:133-51.