AN ENHANCED FREQUENT PATTERN GROWTH BASED ON MAPREDUCE FOR MINING ASSOCIATION RULES

ARKAN A. G. AL-HAMODI[1], SONGFENG LU[*2], YAHYA E. A. AL-SALHI[1]

[1]Research Scholar, School of Computer Science,
Huazhong University of Science and Technology, Wuhan 430074, PR China
[2]Associate Professor, School of Computer Science,
Huazhong University of Science and Technology, Wuhan 430074, PR China

ABSTRACT

In mining frequent itemsets, one of most important algorithm is FP-growth. FP-growth proposes an algorithm to compress information needed for mining frequent itemsets in FP-tree and recursively constructs FP-trees to find all frequent itemsets. In this paper, we propose the EFP-growth (enhanced FP-growth) algorithm to achieve the quality of FP-growth. Our proposed method implemented the EFP-Growth based on MapReduce framework using Hadoop approach. New method has high achieving performance compared with the basic FP-Growth. The EFP-growth it can work with the large datasets to discovery frequent patterns in a transaction database. Based on our method, the execution time under different minimum supports is decreased..

KEYWORDS

Association Rule, frequent pattern, Mapreduce, Hadoop.

1. INTRODUCTION

Frequent pattern [1] mining has been an important subject matter in data mining from many years. A remarkable progress in this field has been made and lots of efficient algorithms have been designed to search frequent patterns in a transactional database (TDB). Discovering frequent itemsets is the computationally intensive step in the task of mining association rules. The main challenge is that the mining often needs to generate a huge number of candidate itemsets. In 1994, the Apriori algorithm is proposed by Agrawal and Srikant to solve the problem of mining frequent itemsets [2]. The FP-growth [3] method proposed by Han et al., which is use the FP-tree to store the frequency information of the transaction database. Without candidate generation, FP-growth uses a frequent divide-and-conquer method and the database projection approach to find the frequent itemsets.

As typical methods of Hadoop Distributed File System (HDFS) and MapReduce parallel programming model provide a new idea for processing big data. In 2008, MapReduce approach of parallel FP-Growth is proposed by Li et al., to find the itemset mining for large-scale data [8]. These methods ignore frequent itemsets mining for massive small files datasets in Hadoop. With

DOI : 10.5121/ijdkp.2016.6202

the arrival of big data era, massive data are growing rapidly. However, in reality, most of the large-scale data are composed of massive small files. Carns et al. depicted a large number of small files related to many applications [6]. Never the less, in the face of massive small files datasets, the constructed FP-tree in Parallel FP-Growth (PFP) algorithm cannot fit into the memory, which often causes problems such as memory overflow and huge communication overhead. A computing efficiency of the Hadoop platform largely depends on the performance of HDFS and MapReduce [13]. Hadoop was designed specifically to process streaming large files, so when dealing with massive small files, there are important limitations.

This paper is organized as follows: A brief introduction about frequent pattern andMapReduce. The related research works are presented in Sec.2. The Preliminaries aboutAssociationRules, FP-Growth algorithm, Map-Reduce and Hadoop are discussed in Sec.3. The proposed scheme was presented in Sec.4. Experimental data and results are presented in Sect.5. The paper concludes with Section.6.

2. RELATED WORK

Association Rule mining one kind of data mining algorithms, it can be classified into two types: FP-Growth algorithm and Apriori algorithm. The FP-growth and FP-tree algorithm is the efficient algorithm and most common for mining frequent patterns. As mentioned in Ref.2 Agrawal et al, they proposed the Apriori algorithm to solve the problem of mining frequent itemsets. It scan database multiple times, and need many of I/O load, it will cause massive candidate set. FP-Growth algorithm proposed by Han, J. et al. [3], to solve the repeatedly of scaning in a transaction database, it only scan database twice. Mining frequent patterns without candidate generation is proposed for storing compressed, crucial information about frequent patterns, and develops an efficient FP-tree based on FP-Growth method. An improved FP-Growth (IFP-Growth) [7] is proposed to improve the performance of FP-Growth. The main features of FP-tree and the address-table are that reduce the need to rebuild conditional trees and facilitate the task of tree construction. Le Zhou et al. [8] proposed a balanced parallel FP-Growth algorithm, which improves performance of the original PFP algorithm by balancing import of the parallel FP-Growth phase. Wang Yong et al. [9] proposed a parallel association rules mining method based on cloud computing is designed and implemented. Parallel Randomized Algorithm (PARMA) proposed by M. Riondato [10], a parallel algorithm for mining quasi-optimal collections of frequent itemsets and association rules in MapReduce. An incremental FP-Growth mining strategy is presented by Xiaoting Wei et al. [11], which is parallelized under MapReduce framework; to improve an algorithm is effective in reducing time of duplicated work. Xueqing Jiang et al. [12] proposed a novel parallelized algorithm called parallelizing improved single pass ordered tree (PISPO) based on the cloud-computing framework Map Reduce.

3. PRELIMINARIES

3.1 Association Rules:

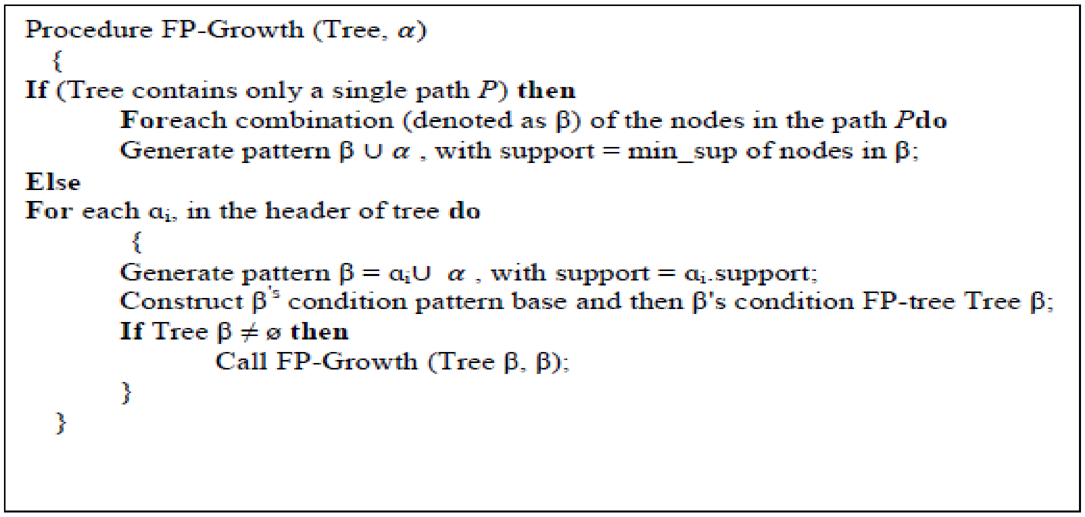
We define $I = \{ I_{1,}I_{2,}I_3, \dots, I_n\}$ as the set of items and n is the number of items. The whole transaction database is defined as $DB = \{T_{1,}T_{2,}T_3, \dots, T_m\}$ and$T_i \subseteq I$. We call the itemset Xcontained in $T_i$only if $X \subseteq T_i$. An association rule ($X \Rightarrow Y$) indicates that$X \subseteq I$, $Y \subseteq I$and at the same time$X \cap Y = \emptyset$. The commonest indicators of association rule strength are support and confidence. sup(X) indicates the frequency of X in database. The confidence of (X ⇒ Y)is defined as,

Confidence (X ⇒ Y) = sup (X ∪ Y)/sup(X).

Association rules mining algorithm aims to search a frequent itemsets meeting user specified minimum support and confidence, then generate association rules needed.

3.2. Basic Fp-Growth:

In data mining, FP-Growth is the most common algorithm used for scanning the patterns in a transaction itemset. To deal with two main drawbacks of Apriori algorithm in [2] a novel, compressed data structure named as FP-tree is constructed, which is prefix-tree structure storing quantifiable information about frequent patterns. Based on FP-tree a frequent pattern growth algorithm was developed into two-step approach. Han et. al. [3] proposed FP-tree mining, is shown in Figure 1. It only scans the database twice without generate candidate set, the algorithm became is very well-known and efficient. The basic idea is scan the transactional database to find frequent 1-item sets, and then construct the FP-tree. FP-Growth discovers conditional pattern base to mine frequent pattern based on the FP-tree. FP-Growth needs to build conditional FP-tree according to each conditional pattern base. Henceforth, the mining process is conducted on independent processes, constructing and mining conditional FP-tree recursively.



```
Procedure FP-Growth (Tree, $\alpha$)
  {
If (Tree contains only a single path $P$) then
        Foreach combination (denoted as $\beta$) of the nodes in the path $P$do
        Generate pattern $\beta \cup \alpha$ , with support = min_sup of nodes in $\beta$;
Else
For each $\alpha_i$, in the header of tree do
         {
        Generate pattern $\beta = \alpha_i \cup \alpha$ , with support = $\alpha_i$.support;
        Construct $\beta^{'s}$ condition pattern base and then $\beta$'s condition FP-tree Tree $\beta$;
        If Tree $\beta \neq \varnothing$ then
                Call FP-Growth (Tree $\beta$, $\beta$);
        }
  }
```

Figure1: Basic FP-tree mining algorithm.

According to Figure 1, there are three features of FP-Growth. First, FP-Growth compresses the entire database into a smaller data structure (FP-tree), resulting in the need to only scan a transaction database twice. Second, it develops a FP-Growth method to prevent the generation of large numbers of candidate itemsets. Third, it generates the conditional FP-tree to mine frequent itemsets, and therefore reduces the search space. According to the previous experimental results, FP-growth algorithm is faster than the Apriori algorithm and several techniques of mining frequent itemsets.

3.3. Map-Reduce

MapReduce [5] is firstly introduced by Google and under MapReduce programming framework we could easily implement parallel algorithm. It consists of two periods, map and reduce. The entire transaction database are divide into sub-partitions and each sub-part split of entire database is sent to a mapper for calculation, each mapper will shuffle its own intermediate output data (key, list (value)) pair to corresponding reducer. Once the reducer has collected all key-value pair, it will begin to run reduce function and calculate the output. Both map andreduce function here are specified and implemented by programmers. MapReduce have two phases are shown in Figure 2:

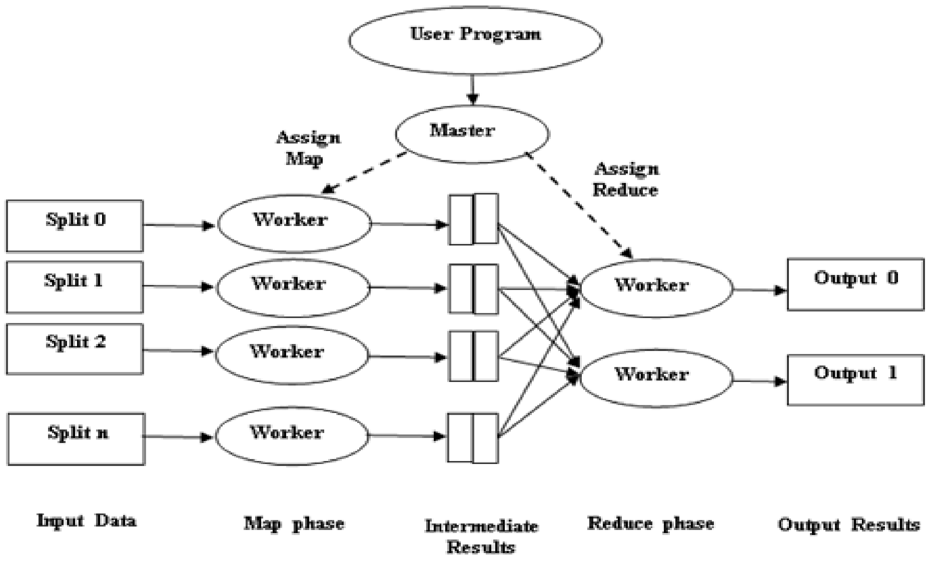


Figure2: MapReduce phases

3.4. Hadoop

Hadoop [14] is a common open source implementation of MapReduce, a strongly tool was designed for deep analysis and transformation of very large data sets which is inspired by Google's MapReduce and File System. It can enable applications to work with many of nodes and petabytes of data. Hadoop can also uses a distributed file system named Hadoop Distributed File System (HDFS), which creates multiple replicas of data blocks and distributes them on compute nodes via a cluster to enable reliable, and has extremely rapid computations to store data such as the intermediate results. Hadoop schedules map and reduce tasks to distributed resources.

4. PROPOSED METHOD

4.1 An enhanced FP-growth (EFP-growth) algorithm

The EFP-growth uses three MapReduce phases to parallelize FP-Growth. Figure 4 shows four steps of EFP-growth as follows.

Step 1: dividing Database (DB) into successive parts (sub-datsets), each part same size according to the transaction numbers, and storing the parts on N different computers nodes. Use Parallel Counting algorithm to count the support values of all items which is appear in DB, and then each sub-datasets will be entered in a single mapper. The result is stored in Frequent list (F-List), Figure 3 show the Parallel Counting algorithm, For each item, $a_i \in T_i$, the a key-value pair will be the outputs of the mapper (key =a_i , value= 1). After all mapper instances have finished, for each key generated by the mappers, the MapReduce infrastructure collects the set of corresponding values and feed the reducers with key-value pairs (key, List (key)).

The reducer thus simply outputs (key = null, value= key + sum (List (key))). It is easy to see that key is an item and value is sup (key).

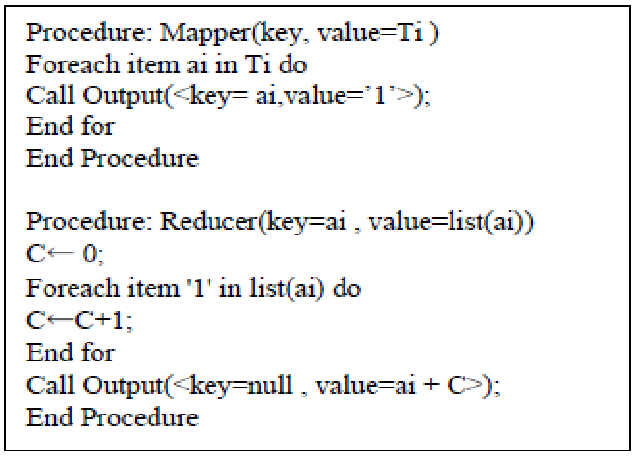
Step 2: Grouping Items: dividing all |I|items into G groups which are arranged in F-List. The final obtained list of groups is called grouplist (G-list), where each group is given an Index F-list (Id). As F-list and G-list are both small and the time complexity is $O(|I|)$, this step can complete on asingle computer in few seconds.

Step 3: Enhanced FP-Growth the main work of EFP-Growth taking one MapReduce pass, where the map and reduce phases are perform different important functions:

Mapper – each mapper instance is arranged with a sub-datasets generated in Step 1. Before it processes transactions in the sub-datasets one by one, it reads the G-list through step 2. The mapper algorithm outputs will be one or more key-value pairs, where each key contains an Id and its corresponding value respected a generated group-dependent transaction.

Reducer – while all mapper instances have finished their work, the MapReduce infrastructure automatically groups all corresponding group-dependent transactions, for each Id, into a sub-datasets of group-dependent transactions. Each reducer instance is assigned to process one or more group-dependent sub-datasets one by one. For each sub-datasets, the reducer instance builds a local FP-tree and EFP-growth its conditional FP-trees recursively. During the recursive process, it may output discovered patterns.

Step 4: aggregating is the results which are generated in Step.3 as last frequent itemsets.



```
Procedure: Mapper(key, value=Ti )
Foreach item ai in Ti do
Call Output(<key= ai,value='1'>);
End for
End Procedure

Procedure: Reducer(key=ai , value=list(ai))
C← 0;
Foreach item '1' in list(ai) do
C←C+1;
End for
Call Output(<key=null , value=ai + C>);
End Procedure
```

Figure 3: parallel Counting Algorithm

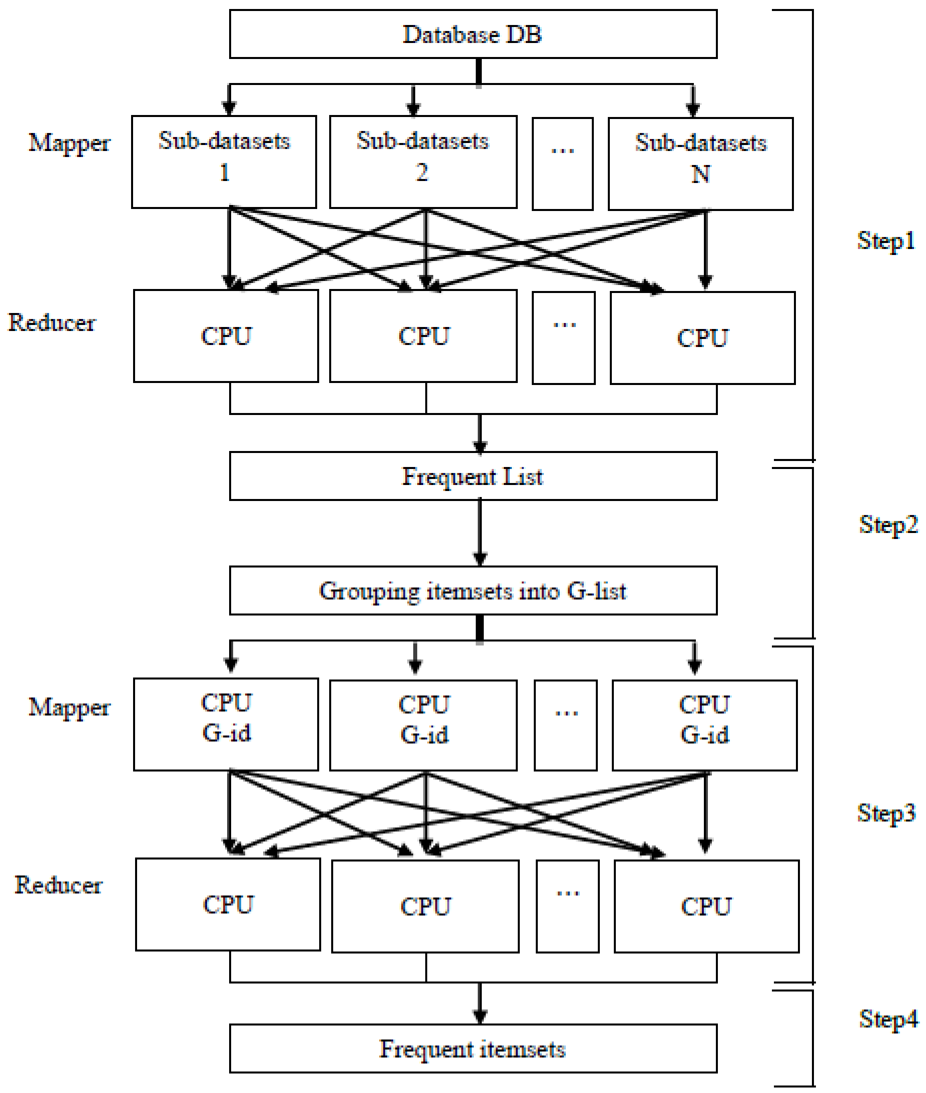


Figure 4: Flow chart of EFP-Growth algorithm

4.2. Implementation of EFP-Growth algorithm based on MapReduce:

4.2.1. Implementation Approach

a) Transaction database is divided into equal-sized sub-transaction database, and is distributed to different computers (nodes) in the cluster.

b) Each node in the cluster separately calculates its own item's support count, and then the results will be aggregated to the same node. The results are arranged by support countdescend, forming a frequent l-itemsets F-list.

c) The required transaction of constructing the frequent item's FP-Tree is distributed to each frequent item corresponding node (It is ensure that the global frequent itemsets are integrated), using classic FP-Growth algorithm to get local frequent itemsets which contain this frequent item.

d) Aggregating local frequent itemsets to obtain the global frequent itemsets.

4.2.2. Implementing Process

The implementation of EFP-Growth algorithm is based on MapReduce and mainly includes four steps as follows. All pseudo codes are omitted here.

Stepl: The TDB is divided into equal-sized sub-transaction database and distributed to different nodes; this step is done automatically by the Hadoop distributed file system HDFS.

Step2: distributed computing F-list.

Step3: Parallel computing local frequent itemsets. Map function takes the frequent item $a_i$corresponding transaction sets and sends them to the same node to form list (FT_i), while Reduce function generates local frequent itemsets by creating and mining condition subtree (SubFP-tree).

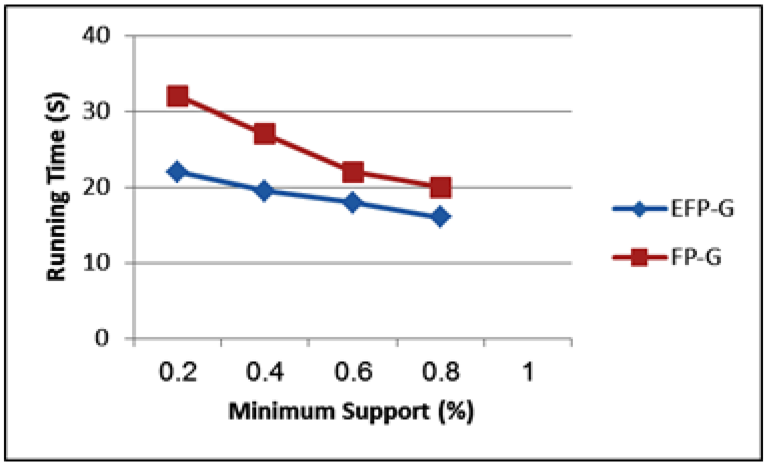
Step4: Local frequent itemsets are aggregated from each node, and then we get the global frequent itemsets.

5. EXPERIMENTAL AND RESULTS

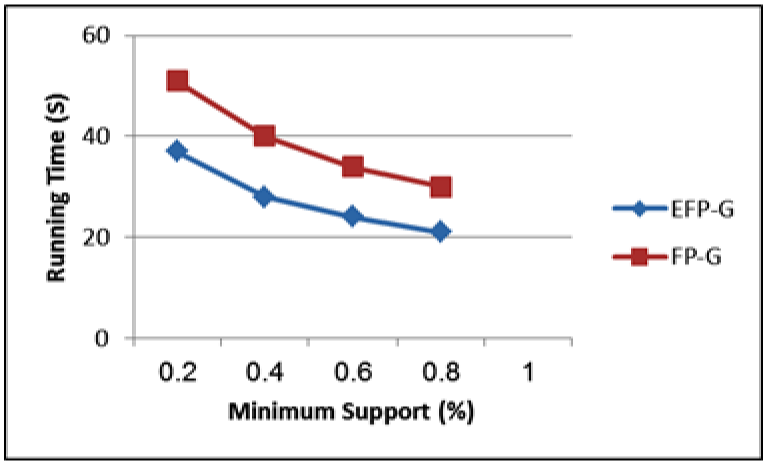
In this section, EFP-Growth and FP-Growth algorithm, were compared and analyzed through experiments. Our results are implemented in Java based on Hadoop environment. The data sets were often used in previous studies of frequent itemsets mining and were downloaded from website http://fimi.cs.helsinki.fi/data/. Table.1 shows the characteristics of classic datasets with the number of items, average transaction length and the number of transactions in each database. Figure 5 gives the running time of FP-Growth and EFP-Growth by using three datasets are connect, T40I1D100k and Kosarak. We can learn from the results that EFP-Growth cost the least amount of time, comparing to the FP-Growth algorithms. When data volume is big, the EFP-Growth shows great advantage in running time over FP-growth algorithms, especially when threshold value is low.

Table 1: Database characteristic

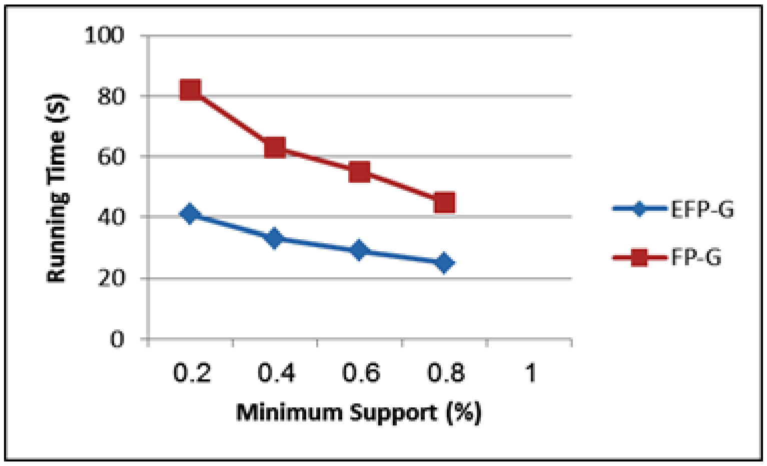
Dataset	**Transaction**	**Items**	**Size(MB)**
Connect	67.557	129	8.82
T40I1D100k	100000	942	14.8
Kosarak	990.002	41270	30.5



(a)



(b)



(c)

Figure 5: Experiment of a: connect; b: T40I1D100k; c: Kosarak

6. CONCLUSION

In this paper, a new scheme called EFP-Growth algorithm is presented. Our scheme is implemented using Hadoop platform under goes MapReduce framework. An associations rules mining with the EFP-Growth has been discussed in this paper. According to the EFP-Growth strategy it can work with the large transaction database for finding the mining frequent itemsets. The results shown that the performance of the new scheme is effective compared with other FP-Growth mining algorithms.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the support from the National Natural Science Foundation of China under (Grant No. 61173050).

REFERENCES

[1] Nasreen, S., et al., (2014) " Frequent Pattern Mining Algorithms for Finding Associated Frequent Patterns for Data Streams: A Survey", Procedia Computer Science,37: p. 109-116.

[2] Agrawal, R., &Srikant, R. (1994). Fast algorithm for mining association rules in largedatabases. In Proceedings of 20th VLDB conference (pp. 487–499).

[3] Han, Jiawei, Jian Pei, and Yiwen Yin, (2000) "Mining frequent patterns without candidate generation." In ACM SIGMOD Record, vol. 29, no. 2, pp. 1-12.ACM.

[4] Li, H., Wang, Y., Zhang, D., Zhang, M. and Chang, E. Y. (2008) “PFP: Parallel FP-Growth for query recommendation,” In: Proceeding of the 2008 ACM conference on Recommender systems, Lausanne, Switzerland, 107-114.

[5] Dean, J. and S. Ghemawat, (2004) "MapReduce: simplified data processing on large clusters", in Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6. 2004, USENIX Association: San Francisco, CA. p. 10-10.

[6] Carns, P., Lang, S., Ross, R., Vilayannur, M., Kunkel, J. and Ludwig, T. (2009) “SmallFile access in parallel file systems,” In: Proceeding of the 2009 IEEE internationalsymposium on parallel and distributed processing, Rome, Italy, 1-11.

[7] Lin, K. C, Liao, I. E. , Chen, Z. S. , (2011) " An improved frequent pattern growth method for mining association rules", Expert Systems with Applications 38 (5), 5154–5161.

[8] Le Zhou; ZhiyongZhong; Jin Chang; Junjie Li; Huang, J.Z.; ShengzhongFeng, (2010) "Balanced parallel FP-Growth with MapReduce," in Information Computing and Telecommunications (YC-ICT), 2010 IEEE Youth Conference on , vol., no., pp.243-246, 28-30 .

[9] Wang Yong; Zhang Zhe; Wang Fang, (2013) "A parallel algorithm of association rules based on cloud computing," in Communications and Networking in China (CHINACOM), 2013 8th International ICST Conference on , vol., no., pp.415-419, 14-16 .

[10] M. Riondato, J. A. DeBrabant, R. Fonseca, and E. Upfal, (2012)"PARMA: AParallel Randomized Algorithm for Approximate Association RulesMining in MapReduce", Proceedings of the 21st ACM internationalconference on Information and knowledge management, pp.85-94.

[11] Xiaoting Wei; Yunlong Ma; Feng Zhang; Min Liu; WeimingShen, (2014) "Incremental FP-Growth mining strategy for dynamic threshold value and database based on MapReduce," in Computer Supported Cooperative Work in Design (CSCWD), Proceedings of the 2014 IEEE 18th International Conference on , vol., no., pp.271-276, 21-23.

[12] Xueqing Jiang; Guozi Sun, (2013) "MapReduce-based frequent itemset mining for analysis of electronic evidence," in Systematic Approaches to Digital Forensic Engineering (SADFE), 2013 Eighth International Workshop on , vol., no., pp.1-6, 21-22 Nov.

[13] White, T. (2012) Hadoop: The Definitive Guide, 3rd ed., O'Reilly Media Inc, Sebastopol,CA.

[14] Shvachko, Konstantin, HairongKuang, Sanjay Radia, and Robert Chansler, (2010) "The hadoop distributed file system." In Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, pp. 1-10.

AUTHORS

ARKAN A. G. AL-HAMODI is a Ph.D student in school of computer science, Huazhong University of Science & Technology, Wuhan, Hubei, P.R. china. He has completed M.Sc from the Department of Computer Science in S.H.I.A.T.SUniversity, India in 2013. His research area of interest includes data mining and information technology. You may contact him at arkan_almalky@yahoo.com

SONGFENG LU is working as an associate professor in School of Computer Science and Technology, Huazhong University of Science and Technology, China. He received Phd in Computer Science from Huazhong University of Science and Technology in 2001. His research areas include quantum computing, information security and data mining. You may contact him at lusongfeng@hust.edu.cn.

YAHYA E. A. ALSALHI is a Ph.D student in school of computer science, Huazhong University of Science & Technology, Wuhan, Hubei, P.R. china. He has completed M.Sc from the Department of Computer Science in B.A.M.U University, Aurangabad, Maharashtra, India in 2012. His research area of interest includes data and information security,digital image processing and algorithm designing. You may contact him at yahya_alsalhi@yahoo.com