

Improved FTWeightedHashT Apriori Algorithm on Big Data using Hadoop-MapReduce Model

ABSTRACT

With the increase of globalization and evolution of Information Technology, a massive amount of data are being generated therefore. Data is expanding dramatically right before our very eyes and it happens to do so at a terrible rate. This trend of explosion of data into Big Data brings forward enormous challenges disguised as incredible business opportunities. The challenge here is that traditional tools are poorly equipped to deal with the scale and complexity of much of this data. That's where Hadoop comes in. Hadoop is used because it provides cheap and reliable storage and tools for analyzing structured, unstructured, and semi-structured information. This explosive growth in stored or temporary data has generated an urgent need for new techniques and automated tools that can intelligently assist us in transforming the vast amounts of data into useful information and knowledge. This has led to the generation of a promising and brilliant frontier in computer science called data mining. Data Mining is the method to gain new models for data analysis in order to get benefits of business. Association rule mining is one of the momentous fields in data mining for generate frequent itemsets and correlations. Frequent itemset mining plays an important role in association rule mining. So-seeming, Mining frequent itemset on big datasets is almost being the most important problem on data mining. Apriori is the best-known basic algorithm for frequent itemset mining. Due to the drawbacks of Apriori algorithm, many improvements were done to make Apriori better, efficient and faster. Weighted based Apriori and Hash Tree based Apriori are the most significant improvements. One of the recent papers integrated the weight concept of weighted Apriori and Hash tree construction concept of Hash Tree Apriori to produce a hybrid Apriori algorithm named WeightedHashT. In this study, we- aim to propose a new approach to improve WeightedHashT Apriori algorithm on big data using Hadoop-MapReduce model. The place and duration of study had been done in Sana'a – Yemen between March 2017 and January 2018. We have reviewed over 100 papers related to our work that include enhancements be done to improve Apriori algorithm. This study used Retail, Mushroom, and T10I4D100K datasets to prove the success of the proposed algorithm. The experimental results indicate that the proposed algorithm has best results for frequent itemset mining in term of execution time. We got 2170 ms whereas time execution of WeightedHashT got 9900 ms. The test is performed on Hadoop cluster consisting of 16 nodes. The present work can be upgraded to higher configuration so that data can be distributed to more number of nodes which will return with better results. A significant performance improvement is possible by introducing a loop-aware scheduler and main-memory caching. These predictors need further work and investigation.

Keywords: [Big Data, Hadoop, MapReduce, and Apriori]

1. INTRODUCTION

We swim in a sea of data and the level of the sea is increasing rapidly. So-seeming, we are actually living in the data age. Terabytes or petabytes of data pours into our computer networks, the World Wide Web (WWW), and various data storage devices every day from business, society, science and engineering, medicine, and almost every other aspect of daily life. This explosive growth of available data volume is a result of the computerization of our society and the fast development of powerful data collection and storage tools. The science of analyzing high volumes of diverse data in near-real time is termed as Big Data [7]. The computerization of our society has substantially enhanced our capabilities for both generating and collecting data from diverse sources. A tremendous amount of data has flooded almost every aspect of our lives. This explosive growth in stored or temporary data has generated an urgent need for new techniques and automated tools that can intelligently assist us in transforming the vast amounts of data into useful information and knowledge. This has led to the generation of a promising and brilliant frontier in computer science called data mining. Data Mining is the method to gain new models for data analysis in order to get benefits of business [4]. Data Mining is also a process of discovering unknown and useful information from large databases. Data mining methods involve seven common classes of data mining techniques according to function and application purpose: outlier detection, association rule mining ARM, classification, clustering, Regression, summarization, and mining time-series data. Among them, the most popular one is association rule mining ARM, which is the process to finding the correlations between items in a given dataset and describe the association relationship among different attributes.

As mentioned above Association Rule Mining (ARM) is the process of finding the correlations between items in a given dataset and describing the association relationship among different attributes. There are many studies have been proposed for association rule mining on big data. The most used algorithm is Apriori. To address the drawbacks of Apriori, number of methods have proposed such as [5] [6] [8- 18]. The common drawback in these algorithms is a delay in execution time. We have noted that the latest study [6] is efficient to improve the performance of Apriori in term of time execution as well as memory utilization; it was designed by combining the benefits of weighted Apriori algorithm and Hash tree Apriori algorithm to get a hybrid Apriori algorithm called "WeightedHashT" whereas, the result that they have introduced is about 50 Mb of memory utilization and about 9900 ms for execution time. With respect to their study, we consider that there is a delay on execution time especially when the data increasing exponentially. Increasing the volume of data with various types; structured, unstructured, and/or semi-structured needs to be processed in fast way by using different architecture than the existed ones. A distributed architecture is able to serve as an umbrella for many different systems. Hadoop is just one example of a framework that can bring together a broad array of tools and services such HDFS and MapReduce for splitting a given big task into small-tasks and running them in parallel way. Thus, the proposed work aims to improve the algorithm presented in [6] by reducing the complexity of time execution and/or memory utilization to get the best result in finding the frequent itemset on big data using Hadoop-MapReduce model.

The proposed work is conducted with boundaries such that, we focused only in generating frequent itemset on big datasets, So generating association rules is not considered. Therefore, will not consider the measure confidence that used in Apriori algorithm.

2. PROPOSED WORK / EXPERIMENTAL DETAILS / METHODOLOGY

The most famous algorithm is Apriori which has been proposed in 1993 by Agrawal [22]. A number of Apriori based algorithms have been proposed to minimize the execution time, reduce the memory consumption, and to enhance the scalability [5] [6] [8-18]. However, [5] [6] are the most recent ones.

[5] Has implemented the Apriori algorithm on MapReduce framework for the central structures hash tree, trie, and hash table trie and evaluated the execution time of Apriori algorithms for these data structures in MapReduce context. This study considers three data structures in Java to be used in MapReduce code. They said "Trie outperforms hash tree experimentally as well as theoretically in sequential computing environment. Initially hash table trie is not a successful in experiments. But trie performs well than hash tree. But hash table trie gives best results in suitable experimental setup and for both real and synthetic datasets." This paper discusses the execution times of three algorithms on dataset T10I4D100K with minimum support of 0.02. The results of their proposed work was 1124s when the chunk size is 100K. However, the authors in [6] have proposed hybrid Apriori association rule Algorithm which combined the benefits of both hash tree and weighted Apriori Algorithms "its tree- based hybrid approach". In their proposed system, the selected synthetic data sets contain numerous numbers of transaction. From these transactions, first, it is needed to predict the items in the transactional dataset. For predicting the frequent items and association rules in the transactional dataset, it generally consumes a lot of memory storage and time. To overcome the problem, their new proposal computes the frequent pattern without the generation of candidate itemsets. Tree based mining approach is used to reduce the consumption of memory usage in the association rule prediction during reducing phase. It uses correlation-based approaches to optimize the process. This work concludes on implantation a new "improved weighted hash tree Apriori Algorithm" which is being termed as hybrid Algorithm in a Hadoop – MapReduce environment and obtained results using a retail dataset and mushroom dataset. The result of this work made sure that the proposed approach works well in terms of time as well as memory. The present work is based on the algorithms mentioned in [5, 6].

This research work aims to improve WeightedHashT [6] Apriori algorithm. Although WeightedHashT succeeded in extracting frequent itemsets, it consumes more time to execute because of dataset multi scans where the scanning process includes frequent and infrequent itemsets. To address this problem, we proposed a novel approach named "FTWeightedHashT". The framework (Figures 1-2) of the proposed algorithm works as follows: a given dataset will be processed in parallel way in Hadoop cluster with n nodes. Part (A) in Figures 1-2 show the different phases of the proposed algorithm for i th node. At the first step, the entire dataset will be stored in a Hadoop. JobTracker will partition the dataset into n small parts ($P_1, P_2, \dots, P_n$). TaskTracker will assign each part P_i ($i=1,2,\dots,n$) to i th node. At the second step, each partition P_i ($i=1,2,\dots,n$), will be scanned (i.e. first scan) to compute the frequency of itemsets (denoted by FP_i). The obtained FP_i is used to compute the weight of itemsets (denoted by WP_i) for transaction filtering as the third step. The way of calculating WP_i is similar to [6] (explained in Chapter 2). Using transaction filtering is sufficient to determine all the frequent itemsets. In other words, if T is a transaction of dataset; then, a filtered transaction of T is the itemsets obtained by removing infrequent itemsets from transaction T (as pruning step). However, transaction filtering was first used in a sequential implementation of Apriori by Christian Borgelt [30]. We apply (at the present work) transaction filtering in efficient parallel implementation of Apriori using Hadoop-MapReduce model. The proposed Transaction filtering phase used to scan the partition P_i at the i th node to determine all frequent itemsets. In fourth step, the obtained frequent itemset will be stored in a file called "FT file" to be scanned instead of scanning the whole partition of dataset. This will address the main drawback of the consumed time calculated by the hybrid WeightedHashT algorithm [6] (i.e. wasting time on scanning both frequent and infrequent itemsets). After that, the tree is constructed by getting the frequent itemsets from the FT file and mapping it to a path of fixed order in the tree (i.e. Second scan) as fifth step. These steps will be repeated until all transactions in i th node are covered to get the frequent itemset for the partition P_i and return back the result to the JobTracker. Certainly, Hadoop will process all the above steps for all nodes in a parallel way and combine the discovered

frequent itemsets from each node and return the final result back to the application as shown in Figure 2.

2.1 Design the Proposed FTWeightedHashT Apriori Algorithm

FTWeightedHashT Apriori algorithm contains three main phases: Item Prediction phase, Filtration phase, and Tree Construction phase (see Figure 3). In the first phase, the algorithm scans the dataset for counting the frequencies of the itemsets. At the second phase, the algorithm calculate the weight of itemsets then filters the itemsets by removing infrequent itemsets (i.e. itemsets with wsupport less than the minimum wsupport value). The last phase contains two sub-phases namely relevancy computation and reachability verification. These sub-phases depicts for constructing frequent tree with direct and indirect relationship using a matrix. The relevancy computation is the process of predicting the relationship between the nodes in the tree which means the itemsets in the frequent itemset tree. The reachability verification refers to the ability to reach the end point of the transaction of each itemset in the tree. We notice that only two passes over the dataset to obtain the frequent itemsets are required (one scan for frequency of itemsets and the other scan to build the tree of frequent itemsets) rather than the algorithm in [6] which take three passes.

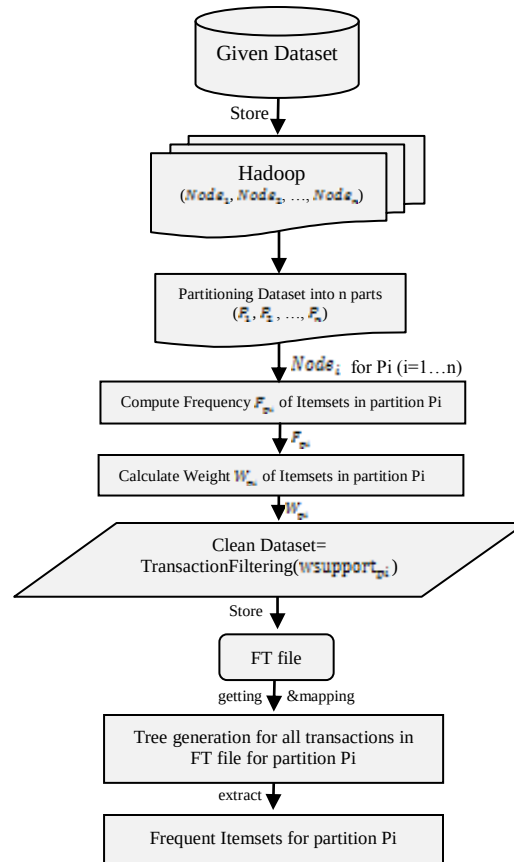


Figure 1: Framework of the Proposed FTWeightedHashT Algorithm

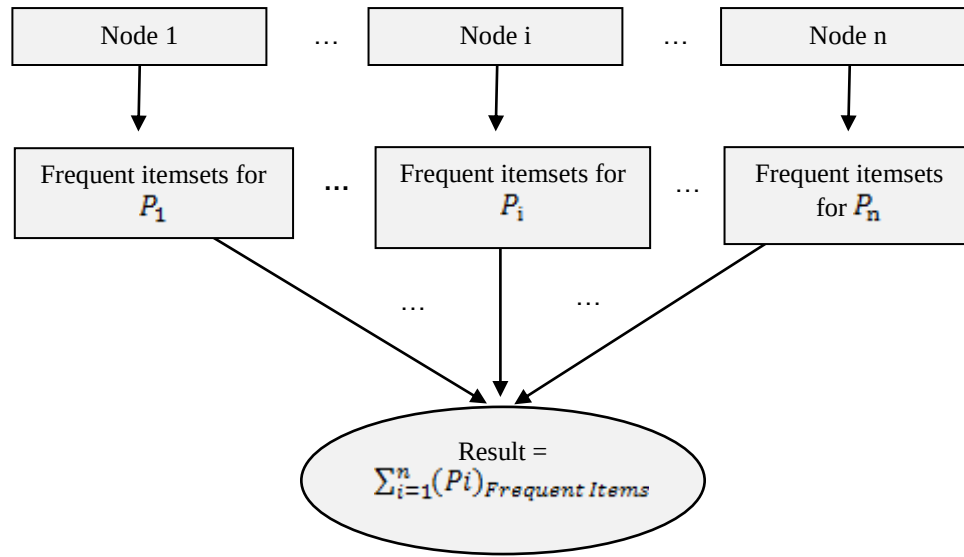


Figure 2: Framework of the Proposed FTWeightedHashT Algorithm for final result

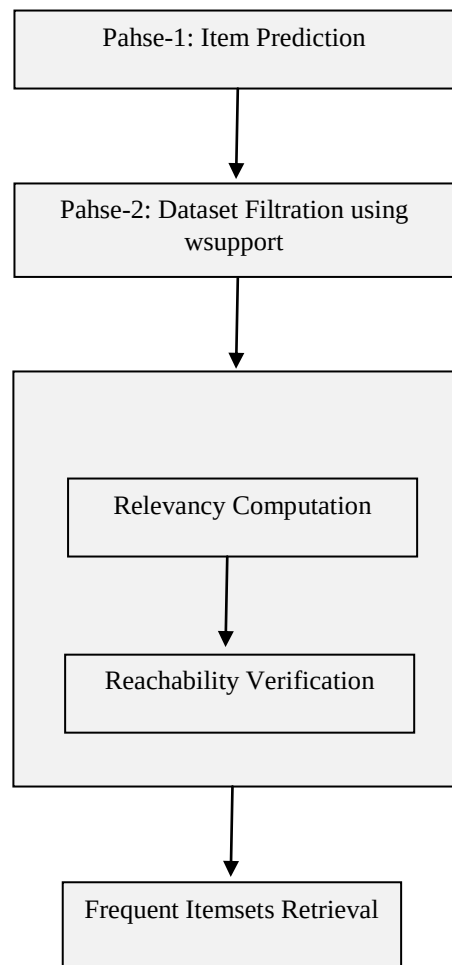


Figure 3: Flow Diagram of the Proposed Algorithm

The proposed algorithm mines the transactional dataset which is considered to be the data for distribution. The algorithm uses weighted support (wsupport) measure instead of using support count measure. The diagram above depicts the way of representing the logic of the algorithm formation. It iterates for specific number of times till no new frequent itemset is found. The algorithm is defined as map tasks to be executed on the distributed data which is available on various clusters of Hadoop nodes. Then the results are combined using reduce task as shown in Figure 2.

2.2 Pseudo Code of the Proposed FTWeightedHashT Apriori Algorithm

The phases presented in the flow diagram of Figure 3 are implemented in Java programming language as given in Figure 4 that is the Pseudo code of the proposed algorithm. The algorithm has a transactional dataset as input and it produces frequent itemsets as output.

```

Input:  $D = \{p_1, p_2, \dots, p_i\}$  where  $p_i$  is the partition of transactional dataset
Output: Frequent Itemsets
FTWeightedHashT Apriori Algorithm
BEGIN
(1)..... TD = Read_TransactionalDataset (D);
(2).....  $F_i = \emptyset$  ;
(3).....  $F_I = \emptyset$  ;
           //Phase-1: Compute Frequency for  $F_{p_i}$ 
(4)..... For  $t = 1 \dots n$  Then //  $n$  = Number of Transactions
(5).....   For  $i = 1 \dots m$  then
(6).....      $F_I = F_I \cup \text{Indexof}(I_i)$ ;
(7).....      $F_{I_i} = F_{I_i} + 1$  ;
(8).....   End For  $i$ 
(9)..... End For  $t$ 
           // Phase-2: Transaction Filtering for
(10).....  $WV_i = \text{Cal\_Weight}(F_i, F_I)$  ;
(11)..... If  $(WV_i < \text{wsupp})$  then
(12).....    $F_i$  . Remove  $(I_i)$ ;
(13)..... End If
           // Phase-3: Tree Construction
(14)..... Tree  $T_r = \text{generateTree}(F_i)$ ;
(15)..... For  $t = 1 \dots n$  Then //  $n$  = Number of Transactions
(16).....   If  $T_r$  .contains( $T_t$ ) then           // Phase-3.1: Relevance Computation
(17).....     For each combination (denoted as CT) of  $T_t$ 
(18).....       Generate pattern CT U RT with  $\text{wsupp}_t \geq \min(\text{wsupp}_t)$  ;
(19).....   Else for each  $N_i \in T_t$            // Phase-3.2: Reachability Verification
(20).....     Header of Tree {
(21).....       Generate pattern CT= $N_i$  U RT with  $\text{wsupp}_i \geq N_i . \text{wsupp}_i$ 
(22).....       Construct CT's conditional Frequent TreeCT;
(23).....       If TreeCT!= 0 then
(24).....         RT = CT;
(25).....       Repeat;
(26).....     }
(27).....   RT = CT;
(28)..... End IF
(29)..... End For  $t$ 
(30)..... End algorithm

```

Figure 4: Pseudo Code of the Proposed Algorithm

3. RESULTS AND DISCUSSION

277

278 This section will provide a description of datasets used in our experiments, Environment
279 setup, and the results of our proposed algorithm.

280

3.1 Dataset Description

282 The improved algorithm is tested with retail, mushroom datasets. One more dataset
283 T10I4100K is also experimented. Retail itemset is having 176324 transactions, 16470 items
284 and with a size of 4153 KB. Mushroom data is with 8124 instances, 22 attributes and with a
285 size of 558 KB. T10I4D100K dataset which is generated by IBM generator consists of
286 100000 transactions, 4 attributes and with a size of 3928 KB.

287

3.2 Environment Setup

288 The experiment is carried out in a machine with Intel core i5-G570 with 2.40GHz and RAM of
289 12GB. VirtualBox workstation is installed. Most of Literature works have used 2 VMs, one VM
290 for Hadoop and the other VM for Eclipse. However, the present work has used only 1 VM for
291 both Hadoop and Eclipse. The VM is used for Hadoop 2.6.0 – cdh 5.10.0 setup and Eclipse
292 3.3.2 with the configuration of 167GB SSD hard disk and 8192 MB base memory. RedHat
293 64-bit is installed in it. Virtual machine are created using VirtualBox workstation 7.1.3.
294 Cloudera-quickstart-vm-5.10.0-0 is the platform which the Hadoop and Eclipse work in.

295

3.3 Results

297 In the following, we will compare the efficiency between FTWeightedHashT, WeightedHashT
298 [6], and HashTableTrie [5]. In Table 1, we are interested in the number of scans, number of
299 iterations, and execution time. The comparison is done for the three algorithms. Compared
300 with former algorithms, the time consumed for FTWeightedHashT is 2170 ms, the number of
301 scans are 2, while the number of iterations are 4. Thus, the execution time for
302 FTWeightedHashT is much less than WeightedHashT and HashTableTrie as shown in
303 Figure 5.

304

305 **Table 1. Comparative Study of the proposed algorithm with other algorithms**

306

Algorithm	Execution Time (ms)	No. of Scans	No. of Iteration
HashTableTrie	29300	Unknown	5
WeightedHashT	9900	3	4
FTWeightedHashT	2170	2	4

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

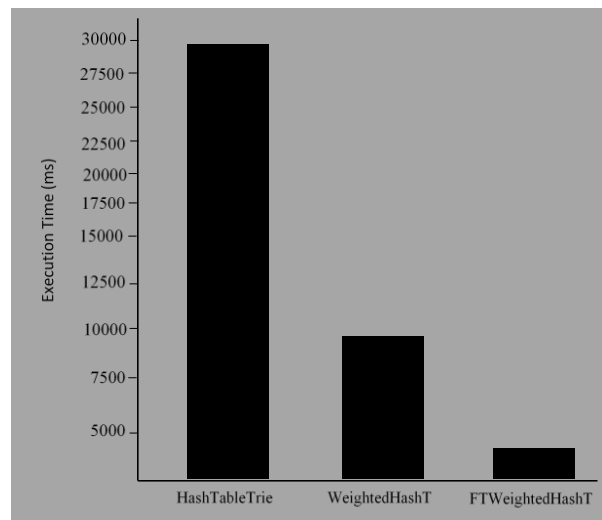


Fig. 5. Execution time Comparison for the proposed algorithm with other algorithms

4. CONCLUSION

Re-designing data mining algorithms on MapReduce paradigm to analyze and mine big data is the new movement in research. So we proposed a novel approach called "FTWeightedHashT" Apriori algorithm for extracting frequent patterns. In our experiment on Hadoop cluster, various comparison charts are given to analyzed the performance and it is made sure that the presented work is the best in term of time and space where the proposed algorithm obligates the process to read only the frequent itemsets from FT file instead of reading in each iteration from the dataset which contains both frequent and infrequent ones. Further possible study could be done in order to improve and enhance of the Hadoop-based Apriori. One can apply the existing idea with auto-multi minimum support because for many datasets it is difficult to choose the value of minimum support. Too small values may cause rule explosion, and too large values may cause rare item deadlock. Thus, implement auto-multi minimum support expected to give better result. Another thing is that we can apply the proposed algorithm to be run on Spark instead of MapReduce. Because of main memory caching mechanism that Spark has, it will give better results absolutely

COMPETING INTERESTS

Declaration of competing interest should be placed here. All authors must disclose any financial and personal relationships with other people or organizations that could inappropriately influence (bias) their work. Examples of potential conflicts of interest include employment, consultancies, honoraria, paid expert testimony, patent applications/registrations, and grants or other funding. If no such declaration has been made by the authors, SDI reserves to assume and write this sentence: "Authors have declared that no competing interests exist.".

REFERENCES

1. Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters, OSDI '04: 6th Symposium on Operating Systems Design and Implementation, pp137-149. (2004)
2. Xin Yue Yang, Zhen Liu, and Yan Fu, Mapreduce as a Programming Model for Association Rules Algorithm on Hadoop. pp 99-102. (2010).
3. Sanjeev Rao, Priyanka Gupta. Implementing Improved Algorithm Over Apriori Data Mining Association Rule Algorithm. IJCST Vol.3(1) pp. 489-493 (2012).
4. Jiawei Han, Micheline Kamber, and Jian Pei, Data Mining: Concepts and Techniques, Elsevier, Third Edition (2013)
5. Sudhakar Singh, Rakhi Garg, P.K.Mishra. Performance Analysis of Apriori Algorithm with Different Data Structures on Hadoop Cluster, Journal of Comouter Applications Vol.128(9) pp.45-51 (Oct-2015).
6. R. Sumithra, Sujni Paul and D. Ponmary Pushpa Latha, A hybrid Algorithm Combining Weighted and Hash T Apriori Algorithm in MapReduce Model Using Eucalyptus Cloud

- Platform, WSEAS Transactions on COMPUTERS Vol.14 . E-ISSN: 2224-2872, pp.382-388 (2015).
7. A.L.Sayath Saabith, Elankovan Sundararajan, Azuraliza Abubaker, PARELLEL IMPLEMENTATION OF APRIORI ALGORITHMS ON THE HADOOP-MAPREDUCE PLATFORM- AN EVALUATION OF LITERTURE, JATIT, Vol.85(3), pp.321-351 (March-2016).
8. Ezhilvathani, A., & Raja, K. Implementation of parallel apriori algorithm on hadoop cluster. International Journal of Computer Science and Mobile Computing. Vol. 2(4), pp. 513-516 (2013).
9. Hazarika, M., & Rahman, M. Mapreduce Based Eclat Algorithm for Association Rule Mining in Datamining : Mr _ Eclat. International Journal of Computer Science and Engineering, Vol. 3(1), pp.19–28. (2014).
10. Itkar, S., & Kulkarni, U. Distributed Algorithm for Frequent Pattern Mining using Hadoop-Map Reduce Framework. Proceeding of International Conference on Advances in Computer Science, AETACS, pp.15-24. (2013).
11. Jyoti, L. D., & Kiran, B. D. A Novel Methodology of Frequent Itemset Mining on Hadoop. International Journal of Emerging Technology and Advanced Engineering, Vol. 4(7), pp. 851–859 (2014).
12. Kovacs, F., & Illes, J. Frequent itemset mining on hadoop., 2013 IEEE 9th International Conference on Computational Cybernetics, pp. 241–245. (2013).
13. Modgi, M. P. Mining Distributed Frequent Itemset with Hadoop. International Journal of Computer Science & Information Technologies(IJCSIT), Vol5(3), pp. 3093– 3097. (2014).
14. Moens, S., Aksehirli, E., & Goethals, B. Frequent Itemset Mining for Big Data. Big Data, 2013 IEEE International Conference on, pp. 111–118. (2013).
15. Oruganti, S., Ding, Q., & Tabrizi, N. Exploring HADOOP as a Platform for Distributed Association Rule Mining. In FUTURE COMPUTING 2013, The Fifth International Conference on Future Computational Technologies and Applications pp. 62–67. (2013).
16. Qureshi, Z., & Bansal, S. Improving Apriori Algorithm to get better performance with Cloud Computing. International Journal of Software & Hardware Research in Engineerin, Vol. 2(2),pp. 33-37. (2014).
17. Vajk, I. Performance Evaluation of Apriori Algorithm on a Hadoop Cluster. Wseas.Us, pp.114–121. (2013).
18. Vinodhini, S., & Vinoth, M. Frequent Pattern Identification using Map Reduce Paradigm. International Journal of Engineering Research & Technology, Vol. 3(3), pp. 1763-1768. (2014).
19. S. Ghemawat, H. Gobio , and S.-T. Leung. The google file system. In ACM SIGOPS operating systems review, volume 37, pp. 29-43. (2003).
20. V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, et al. Apache hadoop yarn: Yet another resource negotiator. In Proceedings of the 4th annual Symposium on Cloud Computing, page 5. ACM. (2013).
21. M. Beyer and D. Laney, “The importance of big data: A definition,” Stamford, CT:Gartner, ID: G00235055.
22. Rakesh Agrawal, Tomasz Imieliński, and Arun Swami, “Mining association rules between sets of items in large databases”, SIGMOD Rec. 22, 2, 207-216. DOI=10.1145/170036.170072. (June 1993).
23. Koliopoulos Aris Kyriakos. A dissertation titled “Big Data Mining: Towards implementing Weka-on-Spark”, The University of Manchester. (2014).
24. <https://users.cs.duke.edu/~maxdml/drupal/?q=distributed-computing-intro>
25. Judith Hurwitz, Alan Nugent, Dr. Fern Halper, and Marcia Kaufman , Big Data For Dummies, Wiley & Sons, Inc., Special Edition (2013)

- 427 26. Google spotlights data center inner workings | Tech news blog - CNET News.com
428 (http://news.cnet.com/8301-10784_3-9955184-7.html)
429 27. [http://www-ai.cs.uni-dortmund.de/auto?self=\\$ejr31cyc](http://www-ai.cs.uni-dortmund.de/auto?self=$ejr31cyc)
430 28. Guy Harrison, Next Generation Databases: NoSQL, NewSQL, and Big Data, APress,
431 First Edition (2016)
432 29. Tom White, Hadoop: The Definitive Guide, O'Reilly, 3rd Edition (2012).
433 30. Christian Borgelt, Efficient Implementations of Apriori and Eclat, In B. Goethals and M. J.
434 Zaki, editors, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining
435 Implementations (FIMI'03), volume 90 of CEUR Workshop Proceedings, Melbourne,
436 Florida, USA, (2003).
437
438
439