Implementation and Analysis of Enhanced Apriori Using MapReduce

Mercy Nyasha Mlambo
Computer Science Department
North-West University
South Africa
nkuzinya@gmail.com

Naison Gasela
Computer Science Department
North-West University
South Africa
Naison.Gasela@nwu.ac.za

Michael Bukohwo Esiefarienrhe
Computer Science Department
North-West University
South Africa
25840525@nwu.ac.za

Abstract— **The exponential growth of data due to advancement in network and computer technologies has driven the need for efficient and real time frequent itemset mining algorithms. Furthermore, as systems are producing large data volumes, there is need for best technologies that can mine and effectively analyze data to obtain crucial information. Association rule mining involves the extraction of associations or connections among data from a given data set. In this research we present market basket analysis for retail stores where mined data associations help marketers to sell frequently purchased combinations of products to their prospective and current customers. Decision makers can also make use of the extracted rules to predict future occurrences and act accordingly. In this paper we present an enhancement of the Apriori algorithm based on a scalable environment called Hadoop MapReduce. Our main goal is to reduce the large resource requirements and minimize communication overheads that are incurred in frequent itemset data extraction using localized split frequent itemset generation and early elimination of infrequent data.**

We also discuss the experimental results obtained from implementing the enhanced parallel Apriori algorithm based on Hadoop MapReduce. Finally we present possible future directions to improve the implementation of Apriori algorithm.

Keywords— Data mining; Apriori; Hadoop MapReduce; Map function.

I. INTRODUCTION

Data mining is the unearthing of useful relationships from huge data using mathematical and computer science technologies. Data mining tools exist for both supervised and unsupervised learning. In supervised learning, a known dataset sample is used to generate training models whilst in unsupervised learning unknown data sets can be employed for training models for data extraction [24]. Data mining techniques can be utilized to generate mining models for specific datasets and can be described based on their function and purpose. Association rule mining is one of the mostly used unsupervised data mining techniques to extract frequent itemsets. The Apriori algorithm has become the most popular but costly association rule mining algorithm utilized since its formulation in 1994 [9].

A. Big Data and Cloud Computing

Due to digitization and technology advancement huge data quantities are generated through a variety of means. Big data involves the analysis of voluminous data using scientific methods taking into consideration the need for real time results. Big data analysis can also be cloud computing based where computer science resources can be shared using available software tools. Big data utilizes both structured and unstructured data formats with the attribute of information variety, volume, veracity and velocity. In which variety shows different data types which can either be complex structured data, unstructured data and semi structured data. Volume attribute determines how large the datasets and its storage facilities and veracity provides data accuracy and fidelity. Cloud computing is a methodology that can encompass parallel and grid computing as well as distributed processing [18]. Cloud computing provides features for resource pooling, network access and virtualization [27]. It is classified into three service models which are as follows [11]:

- Infrastructure as a Service (IaaS) which allows virtualized computer resources to be utilized and the sharing of technology and storage.
- Platform as a Service (PaaS) consists of hardware and software environments which consists of many software tools and allows collaboration software for all users.
- Software as a Service (SaaS) represents the web based application which can be used to provide access rights and performance to applications.

B. Association Rule Mining

Many data mining activities employ techniques for finding frequent itemsets to discover interesting and valuable patterns from big data. These techniques include correlations, association rules, clustering, sequences and classification [10]. Relationships between data can be unearthed by using association rule mining. An association rule in the retail sector may state that the purchase of a certain item/product will result in the customer buying another related product in a retail store. An example is when a customer buys a product X and a product called Y at the same time and they appear on the same receipt. We can say that whenever item X is bought, item Y is also purchased. We can express this example as follows: Buy {X =>Y} where X is the antecedent and Y is the consequent [15]. The rule proposes that a relationship exists between the sale of X and Y due to the manner in which customers are purchasing the goods.

We can measure the strength of association rules by using their support and confidence properties. The support property (Support S) is an expression of how many times a rule can be utilized in a dataset whereas the confidence property (Confidence con) measures the occurrence of Y in the transactions that also have X [15]. Formulas (1) and (2) are used to calculate support and confidence of item X respectively:

$$\text{Support S } (X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{N}, \quad (1)$$

$$\text{Confidence con } (X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}, \quad (2)$$

where N is the number of items X and σ is the support count [9].

C. The Apriori algorithm

Frequent itemset mining is a computational and memory intensive task and has its application in scientific and statistical areas [4]. The datasets keep on growing into big data sources. This means efficient algorithms are required to process these amounts of data. In general candidate itemset generation involves an iterative process which leads to frequent itemset generation being computationally expensive. The following steps are involved in frequent itemset generation.

- Generation of first frequent itemset (1-itemsets): Involves calculation of support count and updating the frequent itemsets as the itemset size increases through the repetitive candidate generation and pruning activities. Given a dataset with an average transaction width of w means the operation of generating frequent 1-itemsets requires O (NW) time (N is the total number of transactions). As a result there is need to employ efficient methods and functions for candidate generation [9, 27].
- Candidate generation: Involves the joining of frequent itemsets to form candidate itemsets. Assuming d items are available, then there will be 2d possible candidate itemsets hence there is a greater need for researchers to come up with efficient methods to generating candidate itemsets [14].
- In the process of generating candidate itemsets (k-itemsets), pairs of frequent itemsets ((k-1)-itemsets) are joined to first itemset and their occurrence I in the transactional database is also verified and counted. The itemset with the highest support is moved into the hash table and then recorded according to its occurrence and support count. Thereafter, all infrequent itemsets are pruned and frequent itemsets undergo joining or candidate generation. As the subset of itemsets becomes large enough the Apriori ends by executing the most frequent itemsets. The produced frequent itemsets are useful in predicting future trends for recommendation to customers as well as packing and arranging grocery in the retail shelf.
  Alternatively, in the worst-case scenario, the Apriori algorithm combines every pair of the most occurring (k-1)-itemsets discovered in the preceding iteration. The resultant cost amount for producing the frequent itemsets is expensive to process. The Apriori algorithm has more challenges when processing big data sequentially compared to when parallel processing.
- Candidate pruning is a process where all infrequent itemsets are deleted from the candidate itemsets. This is an iterative process that eliminates some unnecessarily generated candidate k-itemsets which appear less frequently by using the support-based pruning strategy. Candidate pruning helps reduce input/output cost associated with memory usage during generation of frequent itemsets.

D. MapReduce software framework

The MapReduce (MR) software framework introduced by Google in 2004 is a special tool which is utilized for processing and generating datasets that are extremely large in a parallel manner [26]. Data mining algorithms from association, clustering and classification paradigms have all been explored on MR for data processing. All map and reduce operations are fully parallelizable. The map functions are executed simultaneously and independent from other activities. Similarly, reduce operations can take several datasets and allocate them to each reduce function so that they can be executed in parallel [20].

Hadoop software framework can process large data sets on clusters of computer nodes in a parallel manner. Authors in [17] presented the Apriori algorithm implementation on the Hadoop framework by employing HDFS to generate a duplication of several data blocks and allocating them to different mappers. HDFS automatically schedules mapper and reducer activities and stores the data. Hadoop also has components that handle the problems related to network communication, such as concurrency control and fault tolerance [13, 25]. HDFS is also an instrument to decrease overhead involved in the scheduling of network communication [8].

The Hadoop is designed to store both structured and unstructured data sets reliably with high bandwidth applications [5]. The resource can grow with demand while remaining economical at every size because of its distributed storage and computing across several nodes or servers. Files in HDFS are divided into splits, typically 64MB, and each block is stored in a local file system. The HDFS implements the NameNode which is responsible for maintaining the HDFS directory tree. This is a centralized service in the cluster operating on a single node [12]. The DataNodes on which those blocks are stored are monitored by the name node [1]. We illustrate the architecture of HDFS in Fig. 1 with NameNode and DataNode.

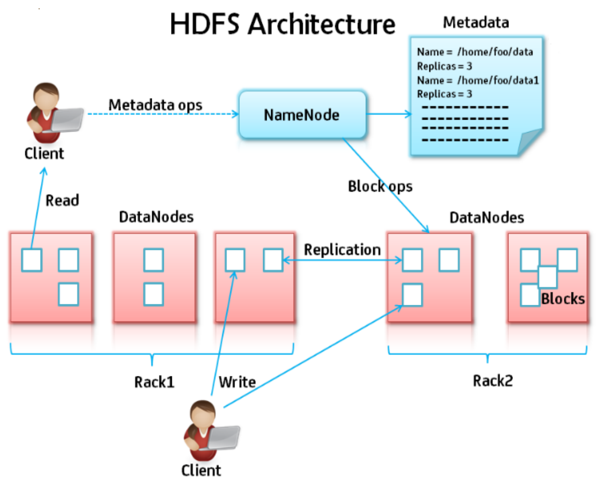


Fig.1. HDFS Architecture[19]

In scenarios where a centralized NameNode is used, all remaining cluster nodes provide the DataNode service where each DataNode stores HDFS blocks [3]. DataNode abstracts details of the local storage arrangement and saves each data block as a separate file when the first data file to be accessed is huge. Other files are manipulated through streaming access patterns similar to those of batch-processing workloads [3].

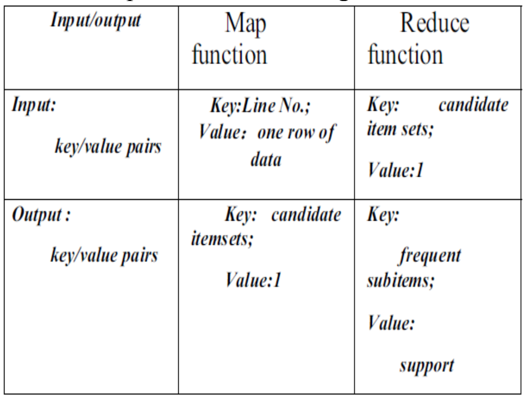
HDFS is divided into large blocks for storage and access, typically 64MB in size. Portions of the file can be stored on different cluster nodes thus balancing storage resources and its availability in case of failure on other storage[26]. Manipulating data at this granularity is efficient because streaming-style applications are likely to read or write the entire block before moving on to the next [6, 16]. In addition, this design choice improves performance by decreasing the amount of metadata that must be tracked in the file system [29].

II. RELATED WORK

New developments in parallel computing have also made data mining easier. Sophisticated multi-core processors and embedded systems have been designed for data mining which decrease processing time and increase performance [28]. The execution of a new multi-core architectural design allowed helper threads to process more blocks of data simultaneously [20]. The designed and developed parallel computers have multi-threading and multi-tasking capabilities which can occur in a single step.
Based on work presented in [7] there are parallel algorithms that can carry multiple tasks concurrently. The algorithms proposed for data mining and embedded systems were to carry out parallel tasks at a lower cost than previous sequential algorithms

The adopted distributed memory structure involves database splitting for each processor that has to be carried out in a logical way so that data can be fairly allocated for processing. Data partitioning must also accompany resource management to enable quick processing of the data. Nilesh et al. [18] presented parallel implementation of the Apriori algorithm on quad core processors. Abaya [21] proposed an implementation of Apriori on CUDA invented by NVIDIA where a collection of multiple running threads employ the power of graphics processing units to escalate computer performance.
Researchers in [19] described the concept of classification, association and clustering in their proposed algorithm Preprocessed Apriori for Logical Matching (PALM). The researchers outlined the problem users are faced with when trying to utilize information found on the internet. They used PALM to do web mining based on MapReduce (MR). A solution for parallelizing Apriori algorithm is implemented on MR using concurrent activities in its execution showing the capability of MR to parallel process data [19]. In the paper presented with the aim of improving the Association Rule Mining (ARM) algorithm, authors in [3] utilized the MR model to provide a parallel design methodology and also outlined a simplified platform for application development. MR possesses the mechanism to minimize the communication costs that are due to the synchronization of tasks between nodes or mapper.
Key/value structured data formats can be processed using the map and reduce methods making Hadoop require Linux and commands expertise to process data. MR was used to parallelize the Apriori algorithm and had the data stored in the Hadoop Distributed File System (HDFS) to ensure that all frequent itemsets were mined . According to [23], Hadoop is a multi-structured framework with different functional units for job execution and file staging. There are also several utilities in Hadoop that allow different formats of files to be stored and used. The datasets are automatically divided into segments by the HDFS and a Map function can be executed on each data segment.

The final output that is produced from MR are key / value pairs as shown in Fig. 2



Input/output	Map function	Reduce function
Input: *key/value pairs*	*Key:Line No.; Value: one row of data*	*Key: candidate item sets; Value:1*
Output : *key/value pairs*	*Key: candidate itemsets; Value:1*	*Key: frequent subitems; Value: support*

Fig. 2. Map/Reduce Key-Value Pair Structure[5]

In MR the dataset is first loaded into HDFS and the format changed to suit the required key/value structure. As illustrated in Fig. 1 key/value pairs act as both inputs and outputs to the simple map and reduce functions. It is therefore crucial to note that the output of map functions are inputs to the reduce function. The candidate itemsets are generated and then collected by the mappers whilst the reducer functions sum their support and prune the unnecessary candidate itemsets which are infrequent [22]. Consecutive executions of map and reduce functions result in determination of frequent itemsets.

According to [13] reducing the communication between slave nodes in a distributed Apriori algorithm on MapReduce effectively reduces communication problems and also reduces the overall time on the project.

The discovery of frequent itemsets based on relationships between data in databases using the Apriori algorithm is a cumbersome process. Although the Apriori is an easy to understand algorithm that uses a simple data structure, the computational complexity is high for the whole process [20]. The Apriori algorithm has been revised in many ways by past researchers. We propose a parallel implementation of the Apriori algorithm that utilizes Hadoop MR and Java Eclipse using HDFS. The implementation will be able to delete transactions that do not satisfy minimum frequency count and the value of k candidate itemsets at each level of increment.

III. THE PROPOSED ENHANCED APRIORI ALGORITHM MODEL

Definition 1: C_k is the candidate itemset of size k, and L_k the frequent itemset of k large [8,11].

Definition 2: Suppose that the transaction database is split into N data splits then each item has local item frequency β_N anda support count σ (global support) which is obtained by adding β_N for all data splits as shown in the formula

$$\sigma = \sum_0^N \beta_N \qquad (3)$$

where β_N is the support count of an itemset in the Nth split and N is the number of processors per split. σ is the global support count which is the total support count of an item. The Apriori Principle suggests that a frequent itemset has also all its subsets frequent [18, 27]. This principle makes the pruning process less complex and also makes it possible to prune using either a top down or bottom up method.

We utilize Hadoop HDFS to map the database transactions and split them into N data blocks to ensure that our improvement strategy scans the database exactly once. The splitting of the data blocks is determined by the processors or mappers available. We programmed the improved Apriori algorithm using Java Eclipse and exported into a jar file. We then employed the jar file in Hadoop MR to extract frequent patterns. MR processes the data by using the Map/Reduce function to obtain local item support, β_N and its global support count, σ. We utilized β_N to prune infrequent itemsets on each data split and σ to delete infrequent itemsets in the whole data set. The support counting results are stored in HDFS. Support counting for candidate itemsets generated is simultaneously calculated using the Map function of the MR concept of Hadoop architecture. Pruning of infrequent itemsets is carried out by using a support based pruning strategy that exploits the Apriori principle and anti-monotone property of support measure inherited from the traditional algorithm. All generated candidate itemsets whose support count does not meet the minimum support count threshold are deleted.

We perform candidate generation of the k-itemsets based on the frequent itemset$_{(k-1)}$ found in the previous iteration. The candidate itemsets are composed of a frequent itemset $_{(k-1)}$ and another frequent itemset$_1$, L_1. Henceforth, all frequent itemset$_k$ are part of the candidate itemset$_k$ that will be generated.

A. Proposed Flow of Activities

In Fig. 3 we illustrate and discuss a sequential flow of activities to be followed in implementing the improved Apriori algorithm.

- Collect and generate retail data

Sales data is collected from retail stores and supermarkets sales online and datasheets collected from retail store. The data is then filtered looking at the minimum number of items and the maximum number of transaction items occurring. Collected data is both structured and unstructured.

- Customize and fine tune retail data

The sales data gathered from internet and retail sales data sheets were duplicated. The data sheets are in excel data formats and some were in text format. We loaded some SQL tables through Sqoop as well.

- Data splitting and uploading on Hadoop

Using the fs copy commands of Hadoop text and excel data is uploaded to Hadoop. The data is also replicated using a set replication factor and then Hadoop converts the data into key-value format that can be processed. Data is partitioned based on the number of mappers available.

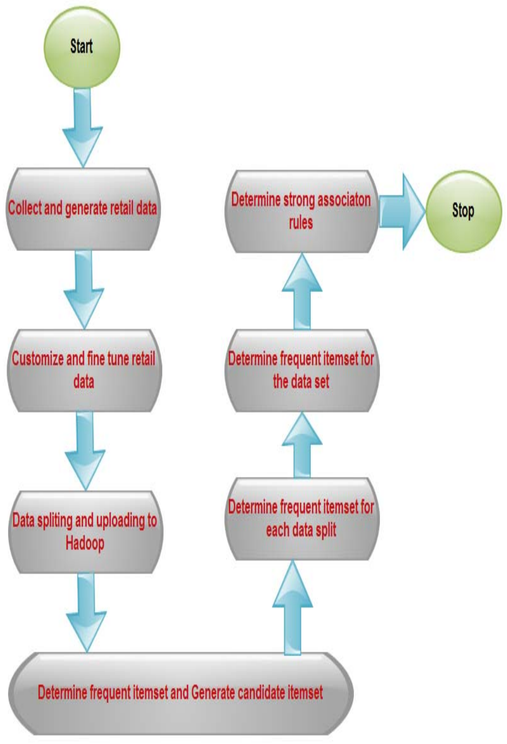

Start
Collect and generate retail data
Customize and fine tune retail data
Data spliting and uploading to Hadoop
Determine frequent itemset and Generate candidate itemset
Determine frequent itemset for each data split
Determine frequent itemset for the data set
Determine strong associaton rules
Stop

Fig. 3. Proposed Flow of Activities

- Determine frequent itemset and candidate generation

Each mapper determines the most frequent itemset and deletes the infrequent items by initially grouping similar products.

- Generate frequent itemset for split

Frequent itemsets are determined using Apriori principle and deleting itemsets that do not satisfy the minimum value of β_N within a data split.

- Determine the frequent itemset for whole data set

Generated frequent itemsets are combined together then, similar itemsets grouped together and then sorted. All itemsets appearing most frequently are considered for the whole dataset.

- Determine the strong association rules

Finally most appearing relationships are recovered from Hadoop and grouped together.

IV. Experimental Setup and Analysis

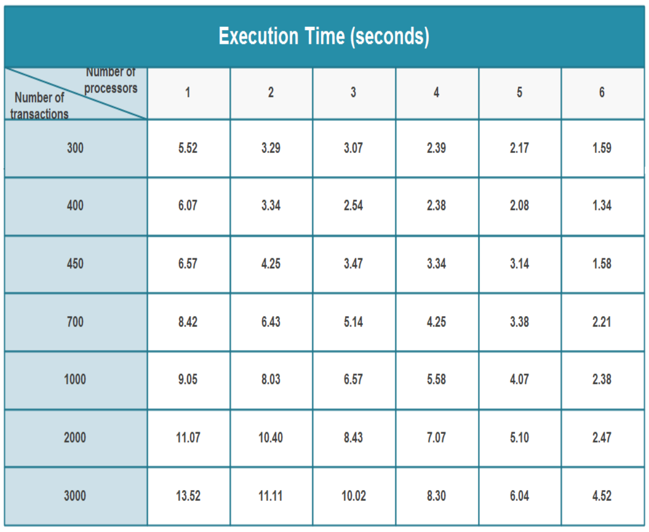
We ran our experiment on Intel® Core™ i7-3770 CPU with a 3.40GHz processor. Eclipse™ IDE (Eclipse Luna service release 2) was utilized to implement the improved Apriori algorithm in the Java programming language. We developed the improved Apriori jar.file that was then executed in Hadoop® MapReduce™. The implementation that was tested on Cloudera™ Hadoop MR Environment had 8 processors running on Windows® 7 Enterprise as the host operating system as well as Linux Redhat® CentOS 6.7 version running on Oracle™ VM Virtual box. The investigation set up had 12 GB of globally accessible memory and 10 GB of accessible memory for the virtual machine. The system was a multiprocessing platform where HDFS shared files and resources with 4 out of 8 CPUs allocated to the virtual box. The Hadoop MR cluster was configured with 16 mappers and 16 reducers. HDFS split data from a range 16 MB to 64 MB chunks presented as a map task with a factor 1. HDFS made duplicate files and stored the data block to ensure that there was data reliability and availability which in turn increased the performance of the algorithm.

The algorithm first extracts frequent itemsets for each split which are then integrated. Then the algorithm determines hidden relationships and discovers frequent itemsets for all splits and eventually generates association rules. The confidence factor is used to determine strong association rules. The map function had to be completed first then the reduce function would start. Fig. 4 shows the progress of the map and reduce functions. When reduce function reaches 100% that means the job has been completed.

B. Results and Evaluation

The execution times obtained from running modified Apriori algorithm on different datasets are recorded in Table 1 below.

TABLE 1 EXECUTION TIMES FOR THE IMPROVED APRIORI ALGORITHM

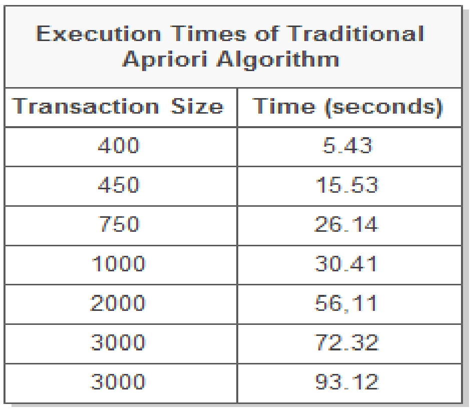


Execution Time (seconds)						
Number of processors / Number of transactions	1	2	3	4	5	6
300	5.52	3.29	3.07	2.39	2.17	1.59
400	6.07	3.34	2.54	2.38	2.08	1.34
450	6.57	4.25	3.47	3.34	3.14	1.58
700	8.42	6.43	5.14	4.25	3.38	2.21
1000	9.05	8.03	6.57	5.58	4.07	2.38
2000	11.07	10.40	8.43	7.07	5.10	2.47
3000	13.52	11.11	10.02	8.30	6.04	4.52

The results shown above are used to analyze and compare

Traditional Apriori algorithm, was also implemented and Table 2 shows the results obtained based on the given transaction size. The execution times are recorded in seconds.

TABLE 2 EXECUTION TIMES FOR TRADITIONAL APRIORI



Execution Times of Traditional Apriori Algorithm	
Transaction Size	Time (seconds)
400	5.43
450	15.53
750	26.14
1000	30.41
2000	56,11
3000	72.32
3000	93.12

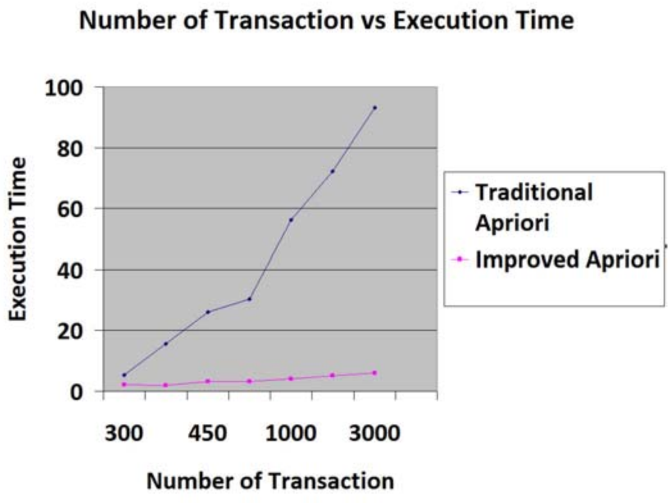


Fig. 4. Number of Transactions versus Execution Time

Fig. 4 shows a comparison of execution times between the traditional Apriori and the Apriori modification. When there are 3000 transactions it takes traditional Apriori 93.13 seconds to process the data, whereas improved Apriori takes only 6.04 seconds to process the same data. The improved Apriori algorithm has a steady increase on time as the number of transactions increases whilst traditional algorithm experiences a sharp increase in execution time as the number of transactions increases. We therefore conclude that localized pruning, support counting and shared execution of activities proposed on improved Apriori algorithm helped enhance the performance of the algorithm.

V. CONCLUSION AND FUTURE WORK

Association rule mining is applicable in many areas and can also be implemented easily. There have been significant improvements and modifications on sequential Apriori algorithms but still the performance of the Apriori in Analysis and predictive systems is still facing memory and performance challenges. We presented a solution where all algorithm processing is centralized at the mappers/nodes and all Apriori algorithm actives are parallelized.

We used a small scale cluster with one master node based on Cloudera Express to evaluate the performance of the improved Apriori. We would want to have our master installed as a Hadoop server that can accommodate many clients/nodes so that we are able to implement a master node - slave nodes architecture. Furthermore, there is a need to fully optimize the Apriori algorithm for market basket analysis where there is no need to input a user specific minimum support and minimum confidence threshold. Further work can also be based on improving parallel and distributed Apriori algorithms based on faster pruning strategies and an efficient candidate generation methods targeting node or mappers level and at the same time devising methods to reduce communication overheads.

REFERENCES

[1] B. Hedlund. (2017). Understanding Hadoop Clusters and the Network [Online]. Available: http://bradhedlund.com

[2] C. Anshu and C. Raghuvanshi, "An algorithm for frequent pattern mining based on apriori," IJCSE) International Journal on Computer Science and Engineering, vol. 2, pp. 942-947, 2010.

[3] C. Zhang and S. Zhang, Association rule mining: models and algorithms: Springer-Verlag, 2002.

[4] F. Masseglia, M. Teisseire, and P. Poncelet, "Sequential Pattern Mining," in Encyclopedia of Data Warehousing and Mining, ed: IGI Global, pp. 1028-1032, 2005.

[5] G. Klockwood, Conceptual Overview of Map-Reduce and Hadoop [Online]. Available: ht tp://www.glennklockwood.com (2017).

[6] H. M. Najadat, M. Al-Maolegi, and B. Arkok, "An improved Apriori algorithm for association rules," International Research Journal of Computer Science and Application, vol. 1, no. 1, pp. 01-08, 2013.

[7] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, Data Mining: Practical machine learning tools and techniques: Morgan Kaufmann, 2016.

[8] J. H. Yeung, C. Tsang, K. H. Tsoi, B. S. Kwan, C. C. Cheung, A. P. Chan, et al., "Map-reduce as a programming model for custom computing machines," in Field-Programmable Custom Computing Machines, 2008. FCCM'08. 16th International Symposium on, pp. 149-159, 2008.

[9] J. Han, J. Pei, and M. Kamber, Data mining: concepts and techniques. Elsevier, 2011.

[10] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," Data mining and knowledge discovery, vol. 8, no. 1, pp. 53-87, 2004.

[11] J. Woo, "Apriori-Map/Reduce Algorithm," in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), p. 1, 2012.

[12] MapReduce, MapReduce Products [Online]. Available: https://mapr.com, (2017).

[13] M. J. Zaki, S. Parthasarathy, M. Ogihara, W. Li, P. Stolorz, and R. Musick, "Parallel algorithms for discovery of association rules," in Scalable High Performance Computing for Knowledge Discovery and Data Mining, ed: Springer, pp. 5-35, 1997.

[14] M. Kantardzic, Data mining: concepts, models, methods, and algorithms: John Wiley & Sons, 2011.

[15] M. H. Awadalla and S. G. El-Far, "Aggregate function based enhanced apriori algorithm for mining association rules," International Journal of Computer Science Issues, vol. 9, no. 3, pp. 277-287, 2012

[16] M. R. Raval, I. J. Rajput, and V. Gupta, "Survey on several improved Apriori algorithms," IOSR Journal of Computer Engineering, pp. 57-61, 2013.

[17] M.-Y. Lin, P.-Y. Lee, and S.-C. Hsueh, "Apriori-based frequent itemset mining algorithms on MapReduce," in Proceedings of the 6th international conference on ubiquitous information management and communication, pp. 76, 2012,.

[18] N. Li, L. Zeng, Q. He, and Z. Shi, "Parallel implementation of apriori algorithm based on mapreduce," in Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), 2012 13th ACIS International Conference on, pp. 236-241: IEEE, 2012,.

[19] N. Gowraj, S. Avireddy, and S. Prabhu, "Palm: Preprocessed apriori for logical matching using map reduce algorithm," International Journal on Computer Science and Engineering, vol. 4, p. 1289, 2012.

[20] R. Agrawal and J. C. Shafer, "Parallel mining of association rules," IEEE Transactions on knowledge and Data Engineering, vol. 8, pp. 962-969, 1996.

[21] S. A. Abaya, "Association rule mining based on Apriori algorithm in minimizing candidate generation," International Journal of Scientific & Engineering Research, vol. 3, pp. 1-4, 2012.

[22] S. Chaudhari, M. Borkhatariya, A. Churi, and M. Bhonsle, "Implementation and Analysis of Improved Apriori Algorithm," databases, vol. 5, 2016

[23] T. White, Hadoop: The definitive guide: " O'Reilly Media, Inc.", 2012.

[24] W. Lu, F.-l. Chung, K. Lai, and L. Zhang, "Recommender system based on scarce information mining," Neural Networks, 2017.

[25] Wikipedia.Data mining [Online]. Available: https://en.wikipedia.org

[26] Y. Chen, F. Li, and J. Fan, "Mining association rules in big data with NGEP," Cluster Computing, vol. 18, pp. 577-585, 2015.

[27] Y. Ye and C.-C. Chiang, "A parallel apriori algorithm for frequent itemsets mining," in Software Engineering Research, Management and Applications, 2006. Fourth International Conference on, 2006, pp. 87-94.ed to the IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), 2005.