An Improved Technique Of Extracting Frequent Itemsets From Massive Data Using MapReduce

Prajakta G. Kulkarni [1], Prof. S. R. Khonde [2]
[1, 2]Department Of Compute Engineering, Savitribai Phule Pune University
[1, 2] Modern Education Society's College of Engineering
19, Bund Garden, V.K. Joag Path, Pune – 411001.
[1] prajakta.r999@gmail.com, [2] shraddha.khonde@mescoepune.org

***Abstract*—The mining of frequent itemsets is a basic and essential work in many data mining applications. Frequent itemsets extraction with frequent pattern and rules boosts the applications like Association rule mining, co-relations also in product sale and marketing. In extraction process of frequent itemsets there are number of algorithms used Like FP-growth,E-clat etc. But unfortunately these algorithm are inefficient in distributing and balancing the load, when it come across massive data. Automatic parallelization is also not possible with these algorithms. To defeat these issues of existing algorithms there is need to construct an algorithm which will support the missing features, such as automatically parallelization, balancing and good distribution of data. This paper is focusing on a efficient methodology to extract frequent itemsets with the popular MapReduce approach. This new methodology consist an algorithm which is build using Modified Apriori algorithm,called as Frequent Itemset Mining using Modified Apriori (FIMMA) Technique. This methodology works with three mappers, independently and concurrently by using the decompose strategy. The result of these mappers will be given to the reducers using the hash table method. Reducers gives the top most frequent itemsets.**

Keyword-Association Rules, Frequent item sets, Load balancing, MapReduce, Modified Apriori, FIMMA.

I. INTRODUCTION

Frequent itemset mining is a noteworthy research subject in associations, correlations, classification, sequences and other essential data mining tasks. To find out frequent item sets is one of the basic computational task in association rule mining where Frequent Item-set is gathering of similar items that happens together in numerous transactions. In association rule mining, to discover Frequent Itemset , characterizes the two similar itemsets in which first itemsets has similar itemsets of another. These rules are helpful for finding interesting relationships in the datasets and gives insight to the procedure that generated the data[12]. Now a days there are various information creates from different sources like IT enterprises, administrations, advancements and information. These large information is available with different structures. To deal with such excessive information is exceptionally troublesome because it has millions of transactions of users, products etc. There are number of strategies to discover frequent itemsets from database. These techniques function well on usual datasets, however not appropriate on excessive amount of data. To utilize frequent itemset mining strategy on massive database is very critical task. To accelerate the procedure of FIM is complex and indispensable, because FIM consumes vast significant portion of time to do high computation and input/output intensity.

One of the solution to this issue is to use a new parallel frequent itemsets mining algorithm with MapReduce called as FIMMA (Frequent Itemsets Mining with Modified Apriori) [12]. In this modern era datasets are very large so only sequential FIM algorithms are not able to calculate large database and they failed to analyze data correctly and also they suffer from low performance. Therefore there should have been a technique to deal with this issue. MapReduce can solve these issues with large number of databases across number of clusters. This distributed approach is grouped with FIM to remove the issues of sequential FIM and hence performance can be increased. This method improves the capacity of storage and computation of problem.

FIMMA uses the technique of hash tables in which it stores previous results to compare with new one itemsets. It uses buckets in he hash tables to stored the results. Each buckets stores frequent itemsets with the unique number, therefore no need to scan the data again and again to get the result. FIUT is replaced with FIMMA. In this method, issues will be solved that are present in the existing system like data partitioning, load balancing of data. FIMMA takes less time compared with the traditional Apriori. The working of mappers and reducers is done concurrently to optimize the speed, well balancing the load across various clusters[12].

II. RELATED WORK

AAssociation Rule mining for extracting frequent itemsets from the massive database the authors have presented an issue of discovering the frequent items from very large number of database. The authors found out rules that have minimum transactional support and minimum confidence with new algorithm. This algorithm that carefully estimates the itemsets for one pass as well as for all the data. This pruning technique used to avoid certain itemsets. Hence it gives right Association itemsets from excessive databases[1][10]. The authors have presented three algorithms, those are DPC, FPC, and SPC [2], to examine successful executions of the Apriori algorithm in the MapReduce paradigm. SPC has straight-forward functions and the FPC has static passes consolidated checking functions. DPC algorithm gathers the dataset of various lengths by utilizing dynamic method and it gives better performance. As a result, three algorithms will scale up with respect to the increased dataset[2].

A balanced parallel FP-growth algorithm used with MapReduce". Frequent itemset mining is a very essential part [12]in association rules and numerous other data mining applications. But when as dataset becomes larger step by step, mining algorithms can not to deal with such excessive databases. Therefore BPFP is utilized to balance the load in PFP(an extension of PFP algorithm [1]), which boosts parallelization and automatically this feature enhances execution. BPFP gives greater performance by using PFP's grouping system [3].

FIUT is another technique for mining frequent itemsets. It is exceptionally productive system for frequent itemset mining(FIM) called as Frequent Itemset Ultrametric Tree (FIUT)[4][11]. It has two main phases of scans of database. n the primary stage it computes the support count for all itemsets in a large database. In later stage it applies pruning strategy and gives only frequent itemsets. By the time frequent one itemsets are calculated, stage two will construct small ultrametric trees. These results will be displayed in small ultrametric trees.

Dist-Eclat, BigFIM are two sort of FIM calculation utilized for MapReduce Framework in "Frequent Itemset Mining for Big Data". Dist-Eclat focuses on speed by load adjusting method utilizing k-FIS. BigFIM predominantly concentrates on mixture approach for mining extensive databases[5]. Apriori calculation is additionally used to develop k th FIS. The K th FIS is required to find frequent itemsets in view of the E-clat procedure. Dist-Eclat, BigFIM and k-FIS are used with round robin approach.

To discover frequent itemsets PARMA method is used by authors which is parallel Randomized approach with MapReduce. PARMA has two functions ,first it gathers the arbitrary data samples and another it uses parallel computing approach, to accelerate the mining procedure. This approach void the replication that is particularly expensive. This is done by making number of small arbitrary parts of the data exchanges and after that applies a mining algorithm on every part autonomously[6].

k Nearest Neighbor Joins utilizing MapReduce is a productive approach to exhibit the k-closest Neighbor i. e. KNN join with the well known programming technique MapReduce for large. This expansive information is disseminated on number of datanodes. In this the mappers circulates the information into different information datanodes and the reducers combines the result of the KNN join[7].

In Diagnosing of Heterogeneous Hadoop Cluster the authors focuses on finding small and primary faults in various Hadoop clusters. The faults which are distributed over number of machines that have slowed down as well as to determine the cause for degraded performance. The long term goal is to enable Hadoop schedulers to generate efficient slot solve the purpose schedules even if they are in heterogeneous clusters. Homogeneous and heterogeneous clusters both face the primary faults and errors present in small or large amounts according to the severity of the data affected. SO the schedulers are design In this, schedulers need to schedule fewer tasks on slower nodes, like CPU hogging, I/O hogging[8].

The primary concentration of Self-Adjusting Slot Configurations for Homogeneous and Heterogeneous Hadoop Clusters is to increase the resource utilization, to decrease the makespan of multiple tasks. Authors have suggested that a technique which utilizes the map slots [9] proportion and Reduce slots proportion as a adjustable knob. This adjustable knob reduces the makespan of a given set and increases the available resource utilization. These slots are filled up to demonstrate limit of performing the task on every machine. It gradually and increasingly allocates the slots to the map[18] and reduce functions. Also they have invented a new Tumm system which manages the slots for the map and reduce task in Hadoop [9].Managing the map and Reduce functions is the basic step for the dynamic allocation of slots according to the requirement of data. In this paper a substantial frequent itemset mining algorithms and their MapReduce implementations are introduced and investigated. The author provide a solution for porting CD algorithm to MapReduce and introduced and improvement on it. The bases of it that it handles the 2 itemsets in special way therefore it can significantly reduce the response time of the Apriori algorithm[13] [14].

III. SYSTEM OVERVIEW

The existing System is able to do automatic parallelization,well distribution of the data with balancing of data using the MapReduce paradigm. It uses three phases of MapReduce to distribute the excessive amount of data. The FIUT algorithm gives the result by using Ultrametric tree, but it has some data leakage issues and slows down the performance. FIUT generate an item dimensionality reduction issue from available sets. No any well balancing techniques has used, so it will generate high complexity. Proposed system is using FIMMA method that is base on Modified Apriori algorithm to overcome the issues of FIUT algorithm. This reduces the time required to scan the transactions. It works faster even,the support count is less. This is faster than the traditional Apriori algorithm and shows 79 percent reduced time.

(1) Objectives of Proposed System:

- To develop a DM system using hybrid algorithm which is based on Modified Apriori and FP Growth for frequent itemset mining using these techniques on high dimensional data.
- This system works with heterogeneous cluster nodes.
- Improved performance and accuracy with automatic parallel processing. It takes less time to scan the database.

In this paper three MapReduce phases are used. The input database is given to the mapper of the first phase of Map Reduce. Its output is obtained from reducer, in the form of frequent one itemset. This candidate 1 itemset shows the completion of the first phase of MapReduce [12]. Second phase accepts the input from first phase in the form of frequent 1- itemsets. Second phase of MapReduce scans all the data to give frequent k-itemsets. It applies pruning technique, to discover frequent and infrequent itemsets. The result is obtained from reducer , in the form of k- frequent itemsets.

IV. SYSTEM ARCHITECTURE

In Proposed System a new data partitioning method is developed to well balance computing load among the cluster nodes.The architecture of the proposed system is as shown in fig.4. Architecture is divided into two main parts.

1. HDFS framework
2. Mapreduce Approch.

HDFS is a hadoop distributed file system and it stores the log files required for HDFS. HDFS framework accepts the data from user. This is done by Jobtracker. Jobtracker destributes the jobs to the datanodes. Each datanode accepts the job from the Namenode, computes it using MapReduce parallel mining paradigm. Final results will again sent to the Namenode in the reduced form.

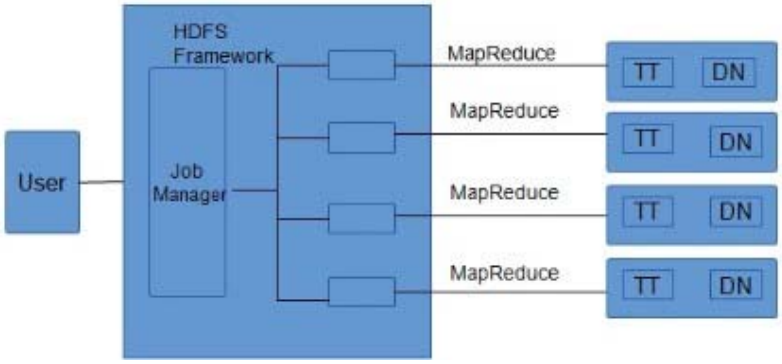

Fig. 1 System Architecture

(2) Detail Internal Architecture flow:

Following fig. 2 shows the internal architecture for the MapReduce jobs. When the jobs distributed among the datanodes ,the working of mappers and reducers will be as follows: When dataset upload is done the first MapReduce accepts the dataset. All dataset will be split into available mappers and the output is combined in the form of frequent 1 itemsets from the reducer. This output will be in the form of key and values.

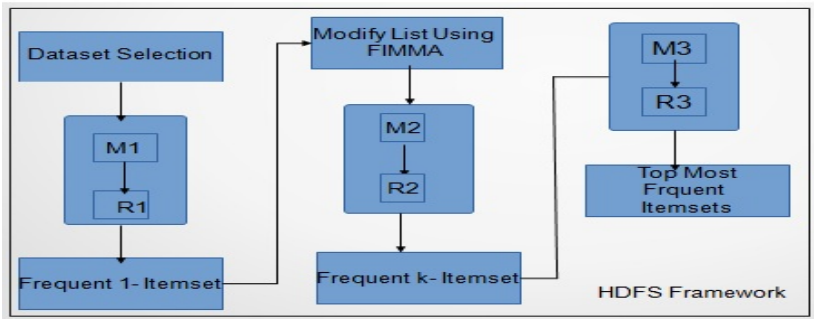

Fig. 2. Detail Architecture flow

Second MapReduce job starts by accepting frequent 1 itemset as shown in above fig. This job scans the frequent 1 itemset again and gives the result in k itemsets. This k itemsets value is set by user. K- frequent itemsets consist all combination of ith itemset. k- frequent itemsets applies to the third MapReduce. Here it gives top most k itemsets from the reducers. FIMMA method uses hash tables with buckets concept. It work from the second MapReduce. It scans all the transaction and makes. Here bucket concept is used to store frequent itemsets. Bucket limit is upto b for each hash table. It will start working from frequent 2 itemsets to frequent k- itemsets. For each bucket a unique value is given. Hash table will store the previous results to avoid unnecessary repetition of frequent itemsets. It raises bit 1 if particular frequent itemset is frequent itemset and raises bit 0 if the frequent itemset is not in the hash table. It uses pruning system and only combine the set vector itemsets.

V. IMPLEMENTATION DETAILS

Three MapReduce phases are used to calculate frequent itemsets. The input dataset is applied to the first MapReduce. This dataset will be accessed using the option ”upload file”. Output is combined from reducer, in the form of frequent-1itemset. This candidate-1 itemset is completed by first MapReduce[12]. Following fig 3 shows the implmentation of frequent-1 itemsets from the given database. Algorithm 1 shows the functionality for the implementation of candidate-1 generation.

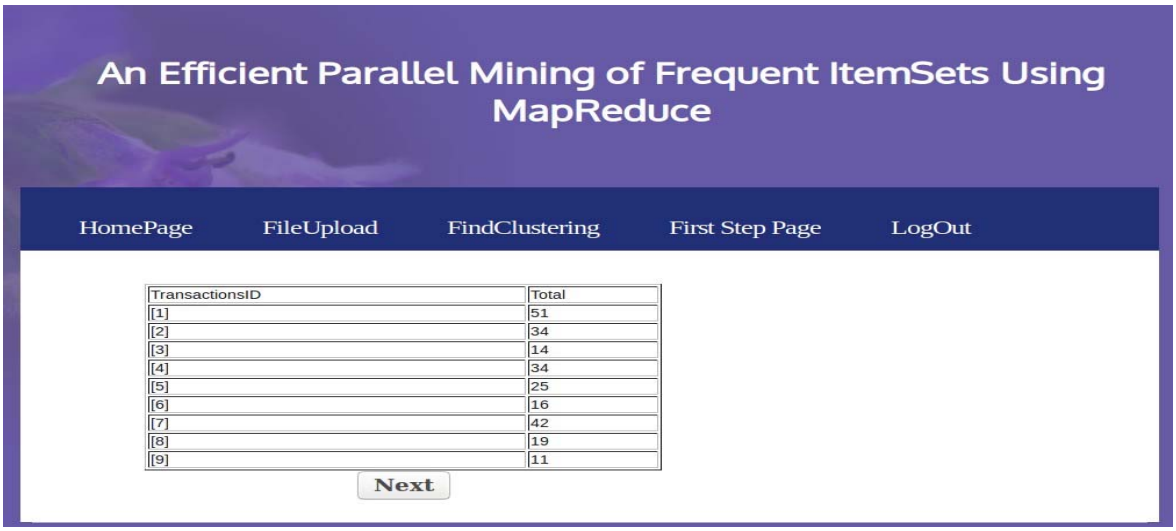

Fig. 3. Implementation of Frequent-1 Itemset

Second MapReduce accepts the input from first MapReduce in the form of frequent 1- itemsets. Here this will scan frequent-1 itemsets again . Infrequent itemsets are removed by using pruning technique, and frequent itemsets will be modified with the help of modified algorithm. The result is obtained from reducer , in the form of k- frequent itemsets. The implementation of frequent k- itemset is shown in following fig. 4.

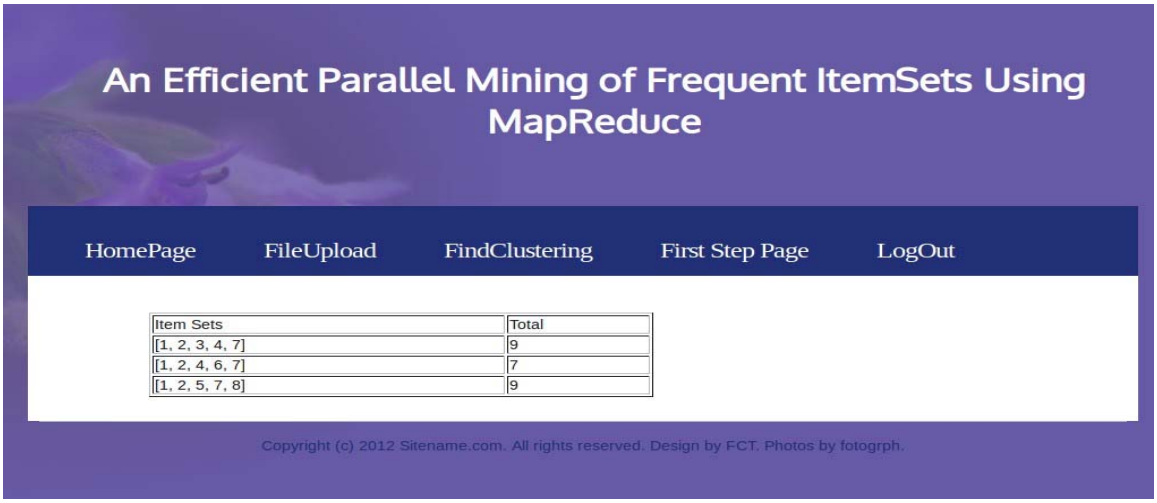

Fig. 4. Implementation of Frequent-k Itemset

VI. METHODOLOG

Following algorithms are used to implementthe frequent-1 itemsets ,k itemset and for the proposed algorithm which is modified algorithm using hashing technique. These are use with map and reduce approach and with minimum support. Minimum support can be given by the user and that will be the k th value [12].

(1) Algorithm for Frequent 1 itemsets.

Input: Dataset Ds, min support Ms.

Output: Freuent 1-itemsets.

Mapper Algo**:**

Mapper(items, DB)

Step 1 : input dataset with k

Step 2: Generate mappers with m maps

Step 3: get all items from transaction T

Step 4: For item in items

Process item as output

End for

Reducer(Key items, Value 1)

Step 1: take all items from list

Each list form each mapper

Step 2: extract relevant set from items

Create countsum for all items

Step 3: if (countsum, minsupport)

Add items into itemset list with transaction

End for[12].

Modified algorithm is used with hashing technique. Hash based apriori accelerates the speed of extracting the itemsets. It generates less number of candidates as this is using hash table to store the previous result. Ther is no need tp scan the dataset again only the user has to compare the new itemsets with the previous result. The algorithm is as shown below:

*(2) **Modified Apriori Algorithm***

Step 1: Input dataset D and set min support

Step 2: Generate items, for all transactions

Step 3: use a function f1 to store frequent itemsets from dataset

Step 4: for each iteam I to n, i=2

Each kth iteration items which having min support

end for

Step 5: create all items which are equals with k itemset list

Step 6: Store the result in a variable fr_output

Step 7: compare the result with new input

Step 8:Modify the list l

Step 9:use if to check the hash tables raised 1 or 0

Step 10: if 1 then add into fr_output else discard

Step 11: fr_output= f1(new itemsets)

Step 12: modify the list k-1 items reach

Step 13: Final top most frequent itemsets.

VII. PERFORMANCE RESULT AND ANALYSIS

The experimental results of this framework with using FIMMA are set to the minimum support. This threshold value is increased slowly with minimum execution cost. This proves that our implementation gives better performance compare to other methods less time. Existing system shows slow execution compared to FIMMA. Through these examinations we have demonstrated that at increased support our implementation produces the same results with improved performance.

The performance of this system against the different frequent itemset mining methods is as shown in fig 5. In existing sysytem implementation tree structure is used to store the results but it requires good programming language or algorithm to give better results within less time and less memory. Accurate results are also depends on the proper selection of algorithms.We could create our own vast dataset against which to likewise run tests, however the cost for doing as such is negligible. The information in the web documents set comes from a real domain and so is meaningful. We have used five sets of data in our experiments. Three of these sets are synthetic data T10I4D100K, T25I10D10K, and T40I10D100k. The other two datasets are real data (Mushroom and Connect-4 data) which are dense in long frequent patterns.

TABLE I TIME REQUIRED FOR FINDING THE FI FROM DATASET

Size of Dataset (Number of Records)	Process Time (seconds)		
	Apriori	FIUT	Modified Apriori(FIMMA)
1000	156	123	105
2000	189	170	126
3000	255	220	165
5000	283	260	223
10000	351	325	285

Table I is showing time required for three methodolofies. Figure 5 shows the comparision graph for three methodologogies. x–axis shows the methodology and y-axis shows the time required to complete the job.

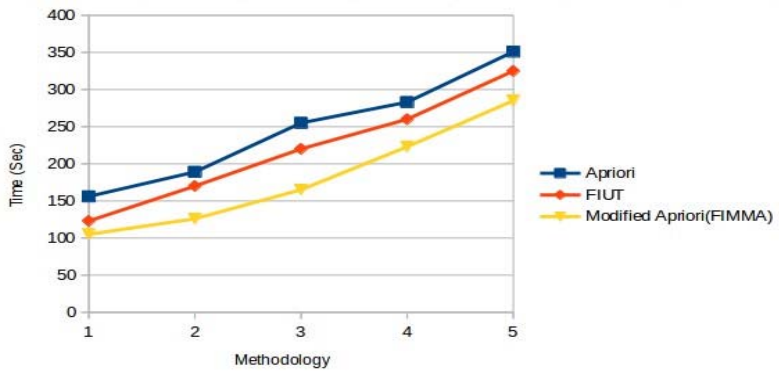

Fig. 5. Comparison graph between different methodologies.

Fig 5 shows the result analysis between Modified Apriori with previous methods in Hadoop environment. By using Hadoop and MapReduce approach it clearly shows that Hash based Apriori took less time compared with other two methods.

VIII. CONCLUSION AND FUTURE SCOPE

To solve the timing issue for the execution of large datasets on Hadoop systems and load balancing challenges in the existing parallel mining algorithms for frequent itemsets, we executed the MapReduce programming model to develop a parallel frequent itemsets mining algorithm called FIMMA. Using the bucket concept in the hash tables, the counter set for the bit vector would increase by 1 when encountered with frequent itemset mining. If the bucket count is equal to or above the minimum support count the bit vector is set to 1 otherwise it is set to 0.Final Result contains the modified list consisting of modified frequent itemsets. By using the Hash based technique in FIMMA it shows the 60 % reduction in time so that we can achieve the speed to fast access the massive databse.

REFERENCES

[1] R. Agrawal, T. Imieli´nski, and A. Swami, "Mining association rules between sets of items in large databases," ACM SIGMOD Rec., vol.22,no. 2, pp. 207–216, 1993.
[2] M.-Y. Lin, P.-Y. Lee, and S.-C. Hsueh, "Apriori-based frequent itemset mining algorithms on MapReduce," in Proc. 6th Int. Conf. Ubiquit. Inf. Manage. Common. (ICUIMC), Danang, Vietnam, 2012, pp. 76:1–76:8. [Online].
[3] L. Zhou et al., "Balanced parallel FP-growth with MapReduce," in Proc. IEEE Youth Conf. Inf. Compute. Telecommun. (YC-ICT), Beijing, China, 2010, pp.243–246.6.
[4] Y.-J. Tsay, T.-J. Hsu, and J.-R. Yu, "FIUT: A new method for mining frequent itemsets," Inf. Sci., vol. 179, no. 11, pp. 1724–1737, 2009..
[5] Kiran Chavan, Priyanka Kulkarni, Pooja Ghodekar, S. N. Patil," Frequent itemset mining for Big data ", IEEE,Green Computing and Internet of Things (ICGCIoT), pp. 1365–1368, 2015
[6] M. Riondato, J. A. DeBrabant, R. Fonseca, and E. Upfal, "PARMA:A parallel randomized algorithm for approximate association rules mining in MapReduce," in Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.,Maui, HI, USA, pp. 85–94 , 2012.
[7] Wei Lu,Yanyan Shen,Su Chen,Beng Chin Ooi,"Efficient Processing of k Nearest Neighbor Joins using MapReduce"2012 VLDB Endowment 2150-8097/12/06,Vol. 5, No. 10,pp.1016-1027.
[8] Shekhar Gupta, Christian Fritz, Johan de Kleer, and Cees Witteveen, Diagnosing Heterogeneous Hadoop Clusters,2012, 23 rd International Workshop on Principles of Diagnosis
[9] Yi Yao, Jiayin Wang, Bo Sheng, Chiu C. Tan, Ningfang Mi, Self- Adjusting Slot Configurations for Homogeneous and Heterogeneous Hadoop Clusters DOI 10.1109/TCC.2015.2415802, IEEE Transactions on Cloud Computing
[10] He Lijun. "Comparison and Analysis of Algorithms for Association Rules", 2009 First International Workshop on Database Technology and Applications, 04/2009
[11] Yaling Xun, Jifu Zhang, Xiao Qin,FiDoop-Dp:Data Partitioning in Frequent Itemset Mining on Hadoop clusters,2016
[12] Yaling Xun, Jifu Zhang, and Xiao Qin, FiDoop: Parallel Mining of Frequent Itemsets Using MapReduce IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, VOL. 46, NO. 3, MARCH 2016
[13] Ferenc Kov acs, Janos Illes, Frequent itemset mining on Hadoop, ICCC 2013,IEEE 9th International Conference on Computational Cybernetics, July 8-10, 2013, pp. 241-245.
[14] S Deshpande, H Pawar, A Chandras, A Langhe ,Data Parti- tioning in Frequent Itemset Mining on Hadoop Clusters- 2016 irjet.net,https://irjet.net/archives/V3/i11/IRJET-V3I11229.pd