International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 5, Issue 1, No 10, 2017 ISSN: 2348-6600
Reference ID: IJCS-205 PAGE NO: 1268-1279

Sri Vasavi College, Erode Self-Finance Wing 3rd February 2017
National Conference on Computer and Communication NCCC'17
http://www.srivasavi.ac.in/ nccc2017@gmail.com

# RECENT SURVEY OF BIG DATA ANALYTICS FOR MAPREDUCE FREQUENT ITEM MINING

S.Prakash1, M.Inbavel2, Dr.P.Siva Prakasam3
1Ph.D Scholar, 2Ph.D Scholar , 3Associate Professor,
Department of Computer Science,
Sri Vasavi College , Erode- 6384016 , Tamilnadu, India,
1prakashmcagobi@gmail.com, 2inbavel@gmail.com, 3psperode@ yahoo.com

**ABSTRACT-** Frequent Itemset Mining (FIM) is one of the most well known techniques to extract knowledge from data process. The combinatorial explosion of FIM methods become even more problematic when they are applied to Big Data. Fortunately, present improvements in the field of parallel programming already provide good tools to tackle this problem. However, these tools come with their own technical challenges, e.g. balanced data distribution and inter-communication costs. In this paper, we analysis the applicability of FIM techniques on the MapReduce platform. In this paper propose a Confabulation Base Parallel FIM approach called CBP-FIM-DP using the MapReduce programming model. The above mentioned FIM mining algorithms extract from and analyze the historical datasets for decision making. The purpose of Big data mining is to go beyond the usual request-response processing, market basket analysis or uncovering some hidden relationships and implement very large scale parallel data mining algorithm. Comparing with the results derived from mining the conventional datasets, unveiling the huge volume of interconnected heterogeneous big data has the potential to maximize our knowledge in the target domain. In our experiments we show the scalability of our methods.

**Index Terms**—Hadoop, Frequent Item Mining, MapReduce, Parallel Algorithm, CBP-FIM

## I. INTRODUCTION

Frequent Itemset Mining (FIM) has been an essential part of data analysis and data mining. FIM tries to extract information from databases based on frequently occurring events, i.e., an event, or a set of events, is interesting if it occurs frequently in the data, according to a user given minimum frequency threshold. Many techniques have been invented to mine databases for frequent events [1], [1], [3]. These techniques work well in practice on typical datasets, but they are not suitable for truly Big Data.

Applying frequent itemset mining to large databases is problematic. First of all, very large databases do not fit into main memory. In such cases, one solution is to use levelwise breadth first search based algorithms, such as the well known Apriori algorithm [4], where frequency counting is achieved by reading the dataset over and over again for each size of candidate itemsets. Unfortunately, the memory requirements for handling the complete set of candidate itemsets blows up fast and renders Apriori based schemes very inefficient to use on single machines. Secondly, current approaches tend to keep the output and runtime under control by increasing the minimum frequency threshold, automatically reducing the number of candidate and frequent itemsets. However, studies in recommendation systems have shown that itemsets with lower frequencies are more interesting [5]. Therefore, we still see a clear need for methods that can deal with low frequency thresholds in Big Data.

Parallel programming is getting utmost importance to deal with the massive amounts of data, which is produced and consumed every day. Parallel programming architectures and supporting algorithms, can be grouped into two main categories viz. shared memory and distributed (share nothing). On shared memory systems, all processing units can concurrently access a shared memory area. While, distributed systems are composed of processors that have their own internal memories and communicate with each other by passing messages [6]. It is easier to port algorithms to shared memory parallelism, but they are typically not scalable enough [7]. Distributed systems, allow quasi linear scalability for well adapted programs. However, it is not always easy to write or even adapt the programs for distributed systems.

Current algorithms like Apriori are good for the databases that are small in size, but if these algorithms are executed on very large databases in parallel on distributed systems the performance can be improved significantly. Hadoop is an open source distributed framework which is designed based on the Google's Map-reduce programming model [8]. Hadoop is capable of analyzing large amountof data. Hadoop is developed by keeping most of the things in mind like-large dataset, write once read many access models, moving computation is cheaper than moving data etc. Hadoop has its own file system called Hadoop Distributed File system (HDFS) which is capable of running on commodity hardware with high fault tolerance ability. Data replication is one of the important features of HDFS, which ensures data availability and automatic re-execution on multiple node failure. In this paper we have proposed algorithm which will use the power of Hadoop for mining the frequent Itemset.

We propose a Confabulation Base Parallel FIM approach called CBP-FIM-DP using the MapReduce programming model. The key idea of CBP-FIM-DP is to group highly relevant transactions into a data partition for confabulation theory; thus, the number of redundant transactions is significantly slashed. Importantly, we show how to partition and distribute a large dataset across data nodes of a Hadoop cluster to reduce network and computing loads induced by making redundant transactions on remote nodes. CBP-FIM-DP is conducive to speeding up the performance of parallel FIM on clusters

This paper is organized as follows: Section II is for background and literature survey, Section III describes the Problem Statement and application of Hadoop to solve this problem and implementation details of the proposed system whereas section IV has proposed algorithm and analytical discussion and finally Section V concludes this paper.

## II. LITERATURE SURVEY

FIM are focused on load balancing data parallel Frequent Itemset Mining techare equally partitioned and distributed among computing nodes of a cluster. More often than not, the lack of analysis of correlation among data leads to poor data locality. The absence of data collocation increases the data shuffling costs and the network overhead, reducing the effectiveness of data partitioning. In this study, we show that redundant transaction transmission and itemset-mining tasks are likely to be created by inappropriate data partitioning decisions. As a result, data partitioning in FIM affects not only network traffic but also computing loads. Our evidence shows that data partitioning algorithms should pay attention to network and computing loads in addition to the issue of load balancing.

Parallel Frequent Itemset Mining. Datasets in modern data mining applications become excessively large; therefore, improving performance of FIM is a practical way of significantly shortening data mining time of the applications. Unfortunately, sequential FIM algorithms running on a single machine suffer from performance deterioration due to limited computational and storage resources [9][10]. To fill the deep gap between massive amounts of datasets and sequential FIM schemes, we are focusing on parallel FIM algorithms running on clusters.

The MapReduce Programming Model. MapReduce - a highly scalable and fault-tolerant parallel programming model facilitates a framework for processing large scale datasets by exploiting parallelisms among data nodes of a cluster [3][4]. In the realm of big data processing, MapReduce has been adopted to develop parallel data mining algorithms, including Frequent Itemset Mining (e.g., Aprioribased [11][12], FP-Growth-based [13][14], as well as other classicassociation rule mining [15]). Hadoop is an open source implementation of the MapReduce programming model [16]. In this study, we show that Hadoop cluster is an ideal computing framework for mining frequent itemsets over massive and distributed datasets.

Data Partitioning in Hadoop Clusters. In modern distributed systems, execution parallelism is controlled through data partitioning which in turn provides the means necessary to achieve high efficiency and good scalability of distributed execution in a large-scale cluster. Thus, efficient performance of data-parallel computing heavily depends on the effectiveness of data partitioning. Existing data partitioning solutions of FiDoop-DP built in Hadoop aim at balancing computation load by equally distributing data among nodes. However, the correlation between the data is often ignored which will lead to poor data locality, and the data shuffling costs and the network overhead will increase. We develop CBP-FIM-DP, a parallel FIM technique, in which a large transactiondataset is partitioned across a

International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 5, Issue 1, No 10, 2017 ISSN: 2348-6600
Reference ID: IJCS-205 PAGE NO: 1268-1279
Sri Vasavi College, Erode Self-Finance Wing 3rd February 2017
National Conference on Computer and Communication NCCC'17
http://www.srivasavi.ac.in/ nccc2017@gmail.com

Hadoop cluster's data nodes in a way to improve data locality.

## III. FREQUENT ITEM METHODOLOGY

### A. Frequent Itemset Mining

Frequent Itemset Mining is one of the most critical and time-consuming tasks in association rule mining (ARM), an often-used data mining task, provides a strategic resource for decision support by extracting the most important frequent patterns that simultaneously occur in a large transaction database. A typical application of ARM is the famous market basket analysis.

In FIM, support is a measure defined by users. An itemset X has support s if s% of transactions contain the itemset. We denote s = support (X ); the support of the rule X ⇒ Y is support (X∪Y ). Here X and Y are two itemsets, and X∩ Y=∅. The purpose of FIM is to identify all frequent itemsets whose support is greater than the minimum support. The first phase is more challenging and complicated than the second one. Most prior studies are primarily focused on the issue of discovering frequent itemsets.

### B. Mapreduce Framework

MapReduce is a popular data processing paradigm for efficient and fault tolerant workload distribution in large clusters. A MapReduce computation has two phases, namely, the Map phase and the Reduce phase. The Map phase splits an input data into a large number of fragments, which are evenly distributed to Map tasks across a cluster of nodes to process. Each Map task takes in a key-value pair and then generates a set of intermediate key-value pairs. After the MapReduce runtime system groups and sorts all the intermediate values associated with the same intermediate key, the runtime system delivers the intermediate values to Reduce tasks. Each Reduce task takes in all intermediate pairs associated with a particular key and emits a final set of key-value pairs. MapReduce applies the main idea of moving computation towards data, scheduling map tasks to the closest nodes where the input data is stored in order to maximize data locality. Hadoop is one of the most popular MapReduce implementations. Both input and output pairs of a MapReduce application are managed by an underlying Hadoop distributed file system (HDFS [17]). At the heart of HDFS is a single NameNode a master server managing the file system namespace and regulates file accesses.

The Hadoop runtime system establishes two processes called JobTracker and TaskTracker. Job-Tracker is responsible for assigningand scheduling tasks; each askTracker handles mappersor reducers assigned by JobTracker. When Hadoop exhibits an overwhelming development momentum, a new MapReduce programming model Spark attracts researchers' attention [18]. The main abstraction in Spark is a resilient distributed dataset (RDD), which offers good fault tolerance and allows jobs to perform computations in memory on large clusters. Thus, Spark becomes an attractive programming model to iterative MapReduce algorithms. We decide to develop

International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600

http://www.ijcsjournal.com Volume 5, Issue 1, No 10, 2017 ISSN: 2348-6600
Reference ID: IJCS-205 PAGE NO: 1268-1279

Sri Vasavi College, Erode Self-Finance Wing 3rd February 2017
National Conference on Computer and Communication NCCC'17
http://www.srivasavi.ac.in/ nccc2017@gmail.com

FiDoop-DP on Hadoop clusters; in a future study, we plan to extend FiDoop-DP toSpark to gain further performance improvement.

C. Parallel FP-Growth Algorithm

In this existing study, we focus on a FP-Growth algorithm called Parallel FP. FP-Growth efficiently discovers frequent itemsets by constructing and mining a compressed data structure (i.e., FP-tree) rather than an entire database. PFP was designed to address the synchronization issues by partitioning transaction database into independent partitions, because it is guaranteed that each partition contains all the data relevant to the features (or items) of that group. Given a transaction database DB, Fig.3.1 depicts the process flow of Parallel FP-Growth implemented in Mahout. The parallel algorithm consists of four steps, three of whichare MapReduce jobs.

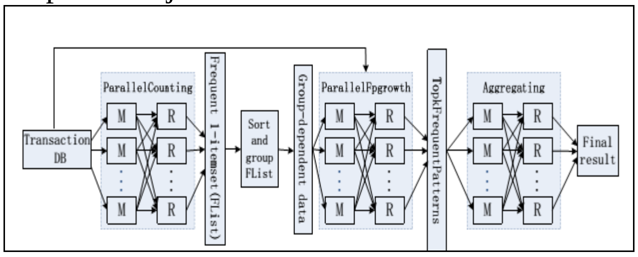

Fig 3.1 Map Reduces

Step 1. Parallel Counting: The first MapReduce job counts the support values of all items residing in the database to discover all frequent items or frequent 1-itemsets in
parallel. It is worth noting that this step simply scans the database once.

Step 2. Sorting frequent 1-itemsets to F List: The second step sorts these frequent 1-itemsets in a decreasing order of frequency; the sorted frequent 1-itemsets are cached in a
list named F List. Step 2 is a non-MapReduce process due to its simplicity as well as the centralized control.

Step 3. Parallel FP-Growth: This is a core step of Pfp, where the map stage and reduce stage perform the following two important functions. Mapper - Grouping items and generating group-dependent transactions and Reducer - FP-Growth on group-dependent partitions. local FPGrowth is conducted to generate local frequent itemsets. Each reducer conducts local FPGrowth by processing one or more group-dependent partition one by one, and discovered patterns are output in the final.

Step 4. Aggregating: The last MapReduce job produces final results by aggregating the output generated in Step 3.

## IV CONFABULATION BASED FREQUENT ITEM METHODOLOGY

### A. Frequent Mining

A major choice in association mining is how the interestingness of an association should be measured. The lift measure has only recently emerged on the scene. Historically, the dominant approach has been support/confidence. In this approach, cells with highest prediction confidence p(i|j) subject to having support above a threshold: p(i,j) > t.

This measure of association arose mainly for computational reasons, since it allows us to restrict our attention to the largest counts, which are easy to identify. (This is not possible with lift, since lift can be high even if the actual count is small, as long as the expected count is even smaller.)

http://www.ijcsjournal.com Volume 5, Issue 1, No 10, 2017 ISSN: 2348-6600
Reference ID: IJCS-205 PAGE NO: 1268-1279

Sri Vasavi College, Erode Self-Finance Wing 3rd February 2017
National Conference on Computer and Communication NCCC'17
http://www.srivasavi.ac.in/ nccc2017@gmail.com

For example, the information that customers who purchase computers also tend to buy financial management software at the same time is represented in association Rule below.
computer ->financial management software [support = 2%; confidence = 60%]
Rule support and confidence are two measures of rule interestingness they respectively reflect the usefulness and certainty of discovered rules. A support of 2% for association Rule means that 2% of all the transactions under analysis show that computer and financial management software are purchased together. A confidence of 60% means that 60% of the customers purchased a computer also bought the software. Typically, association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold. Such thresholds can be set by users or domain experts

B. DIC Algorithm
It is an extension to Apriori algorithm used to reduce number of scans on the dataset.
- Alternative to Apriori Itemset Generation
- Itemsets are dynamically added and deleted as transactions are read
- Relies on the fact that for an itemset to be frequent, all of its subsets must also be frequent, so we only examine those itemsets whose subsets are all frequent.

A dynamic itemset counting technique was proposed in which the database is partitioned into blocks marked by start points. In this variation, new candidate itemsets can be added at any start point, unlike in Apriori, which determines new candidate itemsets only immediately prior to each complete database scan. The technique is dynamic in that it estimates the support of all of the itemsets that have been counted so far, adding new candidate itemsets if all of their subsets are estimated to be frequent.
The resulting algorithm requires two database scans Itemsets are marked in four different ways as they are counted:
- Solid box: confirmed frequent itemset - an itemset the finished counting and exceeds the support threshold minsupp
- Solid circle: confirmed infrequent itemset – the finished counting and it is below minsupp
- Dashed box: suspected frequent itemset - an itemset we are still counting that exceeds minsupp
- Dashed circle: suspected infrequent itemset - an itemset are still counting that is below minsupp

Algorithm Step
- The empty itemset is marked with solid box. All the 1-itemsets are marked with dashed circles. All other itemsets are unmarked.
- Read M transactions. We experiment with values of M ranging from 100 to 10000. For each transaction increment the respective counters for the itemsets marked with dashes.
- If a dashed circle has a count that exceeds the support threshold, turn it into a dashed square. If any immediate superset of it has all of its subsets as solid or a dashed square, add new counter for it and make it a dashed circle.
- If a dashed itemset has been counted through all the transactions, make it solid and stop counting it.
- If the end of the transaction file, rewind to the beginning.
- If any dashed itemsets remain, go to step 2.

After just few passes over the data (usually less than two for small values of M) it finishes counting all the itemsets. Ideally we want M to be as small as possible so we can start counting itemsets very early in step 3.

There are a number of benefits to DIC. The main one is performance. If the data is fairly homogeneous throughout the file and the interval M is reasonably small, this algorithm generally makes on the order of two passes. But if the data is not fairly homogeneous, we can run through if in a random order.

C. Fast-Update DIC

There are many algorithms available for association rule mining. Apriori is the basic algorithm for mining association rules. The Dynamic Itemset Counting (DIC) is an improvement of Apriori.

The FUP (Fast UPdate) algorithm was introduced to deal with insertion of new transaction data. The problem with incremental updating is to find the large itemsets for a database D union db, where D and db are sets of old and inserted transactions respectively. The main assumption is that the set of large itemsets L for D is already known.

FUP is based on the Apriori algorithm. For each iteration, only db is scanned using the known set of large itemsets of size k, Lk, from D as the candidates. This is used to remove the candidates which are no longer large in the larger database, D union db. Simultaneously a set of new candidates is determined. Since the database may change in different ways, FUP can only handle insertion of new transaction data.

FUP2 can efficiently update discovered association rules with insertion of some new transactions and deletion of some obsolete transactions. Both FUP and FUP2 have disadvantages; they require space to store the large itemsets and rules of the original database. In FUP2 the deleted transaction must also be retained and FUP2 is efficient only when the database does not change much.

Another approach to maintain association rules is based on the idea of sampling. The algorithm uses sampling to estimate the upper bound on the difference between the old and new sets of association rules. Small changes to the association rule set are ignored. Motivated by the high number of database scans required by Apriori based algorithms, Partition algorithm was proposed.

In most cases, Partition algorithm requires two complete data scan to mine frequent itemsets. The Partition algorithm divides the dataset into many subsets and each subset can be fitted into the main memory. The main idea of Partition algorithm is that a frequent itemset must be frequent in at least one of the subsets.

During the first data scan, Partition algorithm generates local frequent itemsets for each partition. Since the whole partition can be fitted into the main memory, the complete local frequent itemsets can be mined without further disk access. The local frequent itemsets are added to the global candidate frequent itemsets.

In the second data scan, false candidates are removed from the global candidate frequent itemsets. In a special case where each subset contains identical local frequent itemsets, Partition

International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 5, Issue 1, No 10, 2017 ISSN: 2348-6600
Reference ID: IJCS-205 PAGE NO: 1268-1279
Sri Vasavi College, Erode Self-Finance Wing 3rd February 2017
National Conference on Computer and Communication NCCC'17
http://www.srivasavi.ac.in/ nccc2017@gmail.com

algorithm can mine all frequent itemsets with a single data scan. However, when the data is distributed unevenly across different partitions, Partition algorithm may generate a lot of false candidates from a small number of partitions

D.CBP-FIM-DP Algorithm

CBP-FIM-DP (Confabulation Base Parallel FIM approach called) is about the mining of association rules from the fuzzy dataset which is now the combination of original and fuzzy Hadoop database. The system will check for the itemsets whether they are large in original Hadoop dataset or in fuzzy Hadoop dataset or in both. For the itemsets from the Hadoop dataset, the system will calculate the support threshold, and the itemsets having the support more than the support threshold will be considered as frequent itemsets.

The proposed system uses the DIC technique on the fuzzy itemset Hadoop database. The support threshold for the original Hadoop database has been already calculated. After the DIC is applied the new steps are added for calculation of frequent itemsets. If some itemset is not frequent in original Hadoop database, but when some transactions items are added to the original Hadoop database there may be possibility that the itemset may become frequent, which is not frequent in original Hadoop database. So those itemsets will be checked when new set of transactions items are added to the original dataset. Finally this system will find the frequent itemsets which is having the support value greater than minimum support threshold from the updated Hadoop dataset.

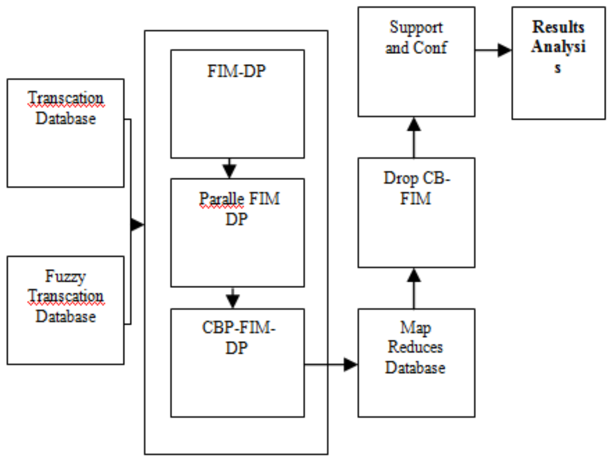

Fig 4.1 CBP-FIM-DP Framework

/* CBP-FIM-DP Algorithm */
Input:
transaction_items: transaction items.
count_items: limit the items scan and finding the frequent items.
temp_items : maintains the temporary frequent or non frequent items.
classify_items: classify the transaction items.
fuzzy_items: If select already set the frequent items or non frequent items.
dynamic_items: increment frequent items.
Output:
association_items: result in frequent item sets (finding minimum support items & update database)
Notations:
ts: transaction items
cu: count the limit items (cuT, cuF, cuD)
temp: temporary frequent or non frequent items.

http://www.ijcsjournal.com Volume 5, Issue 1, No 10, 2017 ISSN: 2348-6600
Reference ID: IJCS-205 PAGE NO: 1268-1279

Sri Vasavi College, Erode Self-Finance Wing 3rd February 2017
National Conference on Computer and Communication NCCC'17
http://www.srivasavi.ac.in/ nccc2017@gmail.com

cs: classify_itemset schema (Classify transaction items)
fs: fuzzy_itemset schema (Classify already set the frequent or non frequent items)
ds: dynamic_itemset schema (finding the increment frequent itemset between fs and cs )
aso: association_itemset schema (finding the frequent itemset in ds)
Method:

```
 Initially state (cs, fs and ds is create and empty the schema)
 Each step finding the frequent items and frequent count (cs, fs, and ds is drop the schema)
Step1: Set the cuT and partition of cuT items in ts.
Step2: Scan and partition the ts and store the cs.
Step3: Set the cuT and partition of cuF items and store in fs.
Step4: Finding the frequent in itemset until ts items between fs and cs.
        If fs is set frequent items means:
            If (items is frequent) means store aso
                Else maintains non frequent items in temp
        If fs is set non frequent items means:
            If (items is non frequent) means store temp
                Else maintains frequent items in aso
        Return   frequent items and frequent count
Step5: The temp item partitions of cuD set and store the ds.
Step 6: Find the increment frequent itemset between cuD to ds.
        If ds is set frequent items means:
            If (items is frequent) means store aso
                Else maintains non frequent items in temp
        If ds is set non frequent items means:
            If (items is non frequent) means store temp
                Else maintains frequent items in aso
        Return   frequent items and frequent count
Step 7: Repeat the process finding frequent items and its counts.
Step 8: Drop the schema fs, cs and ds is each finding frequent items.
Step 9: Finding the association rule mining in minimum support, and confidence following state.
        Support = No. of. Count in Frequent Items / No. of. Total Transaction >= min_support
[i.e. s'    Support(A □ B ) = P(AUB) >= min_support]
        Confidences = No. of. Count in Frequent Items / No. of. Transaction Items >= min_conf
[i.e. s' confidences(A □ B ) = P(B/A) >= min_conf]
        Where min_support: The minimum support threshold.
              min_conf       : The minimum confidences threshold.
Step 10: Update the original Hadoop database is new frequent min_support itemset.
Step 11: Maintains non frequent items future refer the next frequent items.
```

F. Advantages Of CBP-FIM-DP Algorithm

□ Compared to Apriori algorithm and DIC algorithm is less number of update hadoop datasets in database.

□ Scan the Hadoop database easily, because reduces the frequent items and its find previous frequent items and applied the association mining rule algorithm.

- Compare the previous algorithm running speed and time is few reduce in this algorithm.
- Deep level association rule should be applied in this algorithm.
- Effective algorithm and look for a balance between disclosure cost, computation cost and communication cost.
- Efficient and scalable methods for association rule mining should be developed in this algorithm.

## V. EXPERIMENTAL RESULTS

The below Fig 5.1 shows the frequent graph mining for the performance analysis this performance analysis is calculated using frequently searched node sub frequent graph node count and the average mapping frequent sub graph node that is represented as the percentage level.

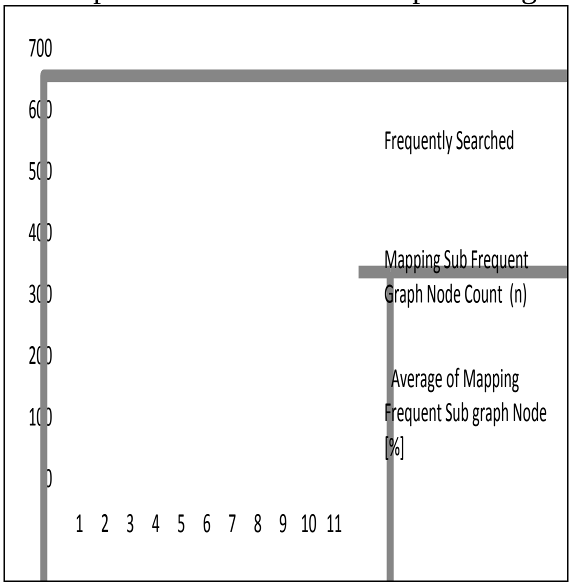

Fig 5.1 Frequent Sub-Graph Mining Performances Analysis

The below Fig 5.2 shows the frequent sub graph node performance analysis that is which node participation in the network is high the average of the mapping is also done. The sub frequent graph node count is been mentioned using the n count value.

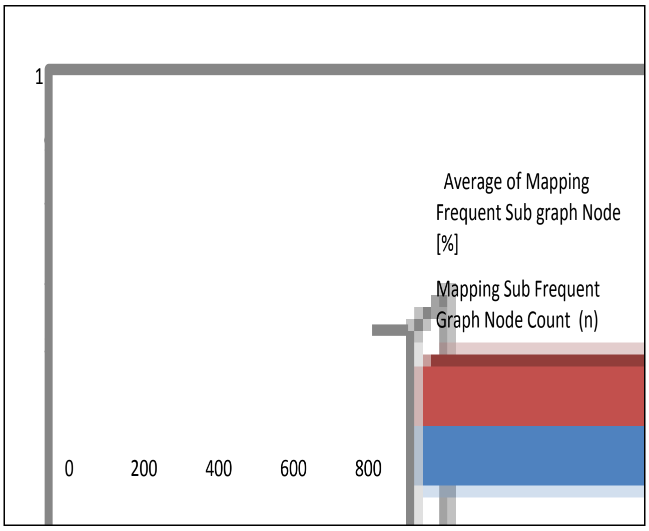

Fig 5.2 Frequent Sub-Graph Mining Performances Analysis(Frequent Sub Graph Node)

## VI. CONCLUSION

In this paper , present the use of an association rule mining driven Hadoop application is to manage mapreduces dataset that provide frequent items with report regarding prediction of product purchase or sales trends and customer behavior. Our goal of the research is to find a new Confabulation based frequent items for finding the rule of the Hadoop dataset, which outperforms in terms of running time, number of database scan,

International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 5, Issue 1, No 10, 2017 ISSN: 2348-6600
Reference ID: IJCS-205 PAGE NO: 1268-1279
Sri Vasavi College, Erode Self-Finance Wing 3rd February 2017
National Conference on Computer and Communication NCCC'17
http://www.srivasavi.ac.in/ nccc2017@gmail.com

memory consumption and the interestingness of the rules over the classical CBP-FIM-DP algorithms.

Hadoop dataset is one of most important part of research process. The main goal of super market industry (Hadoop) sales of frequent items maintains and increasing profit. So there are strong association rule finding frequent items must the data mining works. In this algorithm proposed is some techniques added the future analysis of mining frequent items. Because select the fuzzy items frequent optimizing select and compare transactions items, so strong fuzzy hadoop items created techniques applied our proposed algorithm and update the original hadoop database increasing times, access the speed of processors implementation of our proposed algorithms.

## References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In Proc. VLDB, pages 487–499, 1994.
[2] R. J. Bayardo, Jr. Efficiently mining long patterns from databases. SIGMOD Rec. , pages 85–93, 1998.
[3] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. Parallel algorithms for discovery of association rules. Data Min. and Knowl. Disc. , pages 343–373, 1997.
[4] R. Agrawal and J. Shafer. Parallel mining of association rules. IEEE Trans. Knowl. Data Eng. , pages 962–969, 1996
[5] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule discovery from web usage data. In Proc. WIDM, pages 9–15. ACM, 2001.
[6] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox. Twister: A runtime for iterative MapReduce. In Proc. HPDC, pages 810–818. ACM, 2010.
[7] G. A. Andrews. Foundations of Multithreaded, Parallel, and Distributed Programming. Addison-Wesley, 2000.
[8] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In F. Provost and R. Srikant, editors, Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 401 –406. ACM Press, 2001.
[9] M. J. Zaki, "Parallel and distributed association mining: A survey," Concurrency, IEEE, vol. 7, no. 4, pp. 14–25, 1999.
[10] I. Pramudiono and M. Kitsuregawa, "Fp-tax: Tree structure based generalized association rule mining," in Proceedings of the 9th ACMSIGMOD workshop on Research issues in data mining and knowledge discovery. ACM, 2004, pp. 60–63
[11] 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, ser. ICUIMC '12. New York, NY, USA: ACM, 2012, pp. 76:1–76:8.
[12] X. Lin, "Mr-apriori: Association rules algorithm based on mapreduce," in Software Engineering and Service Science (ICSESS),

2014 5th IEEE International Conference on. IEEE, 2014, pp. 141–144.

[13] L. Zhou, Z. Zhong, J. Chang, J. Li, J. Huang, and S. Feng, "Balanced parallel fp-growth with mapreduce," in Information Computing and Telecommunications (YC-ICT), 2010 IEEE Youth Conference on. IEEE, 2010, pp. 243–246.

[14] S. Hong, Z. Huaxuan, C. Shiping, and H. Chunyan, "The study of improved fp-growth algorithm in mapreduce," in 1st InternationalWorkshop on Cloud Computing and Information Security. Atlantis Press,2013.

[15] M. Riondato, J. A. DeBrabant, R. Fonseca, and E. Upfal, "Parma: a parallel randomized algorithm for approximate association rulesmining in mapreduce," in Proceedings of the 21st ACM internationalconference on Information and knowledge management. ACM, 2012, pp.85–94.

[16] C. Lam, Hadoop in action. Manning Publications Co., 2010

[17] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang, "Pfp: parallel fp-growth for query recommendation," in Proceedings of the 2008 ACMconference on Recommender systems. ACM, 2008, pp. 107–114.

[18] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica,"Spark: cluster computing with working sets," in Proceedings of the2nd USENIX conference on Hot topics in cloud computing, vol. 10, 2010, p. 10.