NOVATEUR PUBLICATIONS
International Journal of Research Publications in Engineering and Technology [IJRPET]
ISSN: 2454-7875
VOLUME 3, ISSUE 3, March-2017

FIDOOP – FIM: DATA SEGREGATION USING FREQUENT ITEM SETS MINING AND MAP REDUCE ALGORITHM

DARSHANA WAJEKAR
Department of Computer Engineering Pillai HOC College of Engineering and Technology
Rasayani, Tal: Khalapur, Dist: Raigad darsh.wajekar90@gmail.com

PROF. EKTA UKEY
Department of Computer Engineering Pillai HOC College of Engineering and Technology
Rasayani, Tal: Khalapur, Dist: Raigad eukey@mes.ac.in

ABSTRACT

Hadoop is an open-source platform of the MapReduce programming model. As data size is increasing gradually day by day, the improvement and security of data structure in Hadoop have become a critical issue. So far, algorithms have lacked of mechanisms like data distribution, fault tolerance, load balancing and input-output overhead. Hence, in order to overcome these discrepancies, the most effective the new method is the FiDoop method using a Map Reduce programming model and FIM algorithm. FiDoop includes the Frequent Item set Ulta metric Tree rather than conventional FP-trees which avoid the necessity to build conditional pattern based on compressed storage. In prior techniques such as pfp, Fidoop and Fidoop-HD the execution time was considerably increasing when the number of records increased. In our proposed system, firstly, the input/output overhead is minimized by scanning the database twice. Secondly, FIUT matric which is an outcome of clustering improves in the partitioning of database and significantly reduce the search space. Later MapReduce plays main act in parallel mining process; mappers separately decay item sets while its reducers make tiny ultra-metric trees to be individually mined. Finally, Fidoop - FIM based on FIM algorithm highly reduces the execution speed of downloading as the number of records increases in size.

KEYWORDS: Hadoop, computing time, Load Balancing, MapReduce, Frequent itemset, Scalability, Fidoop-HD, Fidoop – FIM

I. INTRODUCTION:

Data mining, is a technique to discern and convert raw data into useful information. It is increasingly being used in a variety of fields like scientific discoveries, marketing, Internet searches, multimedia, biotechnology and business intelligence. Data mining is a multidisciplinary field combining ideas from, machine learning, natural language processing and statistics.

Big data is collecting a huge amount of data sets that cannot be processed using traditional computing techniques. In 2005, Mike Cafarella and Doug Cutting provided key by Google and begin an open-source project called HADOOP. Now a day's Apache Hadoop is registered trade name of the Apache Software Foundation. Businesses have greatly benefited from data analytics. Companies are analyzing activities such as sales, stock optimization, advertising and consumer support to improve their strategic, tactical business decision, fraud and risk management. A majority of trendy applications grow to be data-intensive in nature. Illustrative data-intensive web applications consist of on-line auctions, search engines, customizing content for users, supply-chain management, webmail, on-line retail sales, bio informatics, beyond analytics data, fraud analysis, miscellaneous uses, etc.;

Hadoop is an open-source accomplishment of MapReduce, supported by leading IT companies such as Google and Yahoo! Hadoop implements MapReduce framework with two categories of components: a Job Tracker and various Task Trackers. Task Trackers are managed by the Job Tracker and launched on apiece computational node to perform the tasks they receives from Job Tracker. Data processing is performed in parallel through two main tasks: map and reduce. The Job Tracker is in charge of scheduling the map tasks (Map Tasks) and reduce tasks (Reduce Tasks) toward Task Trackers. It also monitors job progress, collects runtime execution statistics, handles possible faults and errors throughout task re-execution. Hadoop merges these intermediary data segments of map task when numbers of data segments go over a threshold. The current merging algorithm frequently merges data segments which cause multiple rounds of disk access for equal data. This degrades the performance of Hadoop.

MapReduce is model for working on distributed computations and huge amount of data. It provides the framework for large scale data processing on commodity hardware. Hadoop distributed file system follows distributed file system and MapReduce is just a programming technique to solve the problem. Mapper and Reducer are the two main components of MapReduce

programming. Each mapper receives pair of key-value as an input and it produces the intermediate arbitrary pairs of key-value as an output. MapReduce is required as it provides abstraction layer to the system-level functions. It just enables developer to perform their operation rather than focusing how to perform that operation.

Big data is a complex task but with help of distributed programming approach it can be resolved. MapReduce is new framework which handles large data using distributed com-putting on large-scale clusters. HDFS a distributed file system stores the input data, MapReduce applies a divide and conquer technique to divide the input data sets into small data sets, and then processed on different machines, which has achieved parallelism. MapReduce follows three step workflow which is Map, Shuffle and Reduce. Data in MapReduce framework is seen as a series of key-value pairs.

In this paper we study about the related work done on the feature selection techniques in section II, the implementation details in section III where we see the system architecture, modules description and algorithms. In section IV we discuss about the expected results and at last we provide a conclusion in section V.

II. RELATED WORK:

In this section, the review of literature relevant to replication merges, disk accesses data and delays the reduce phase are presented. A key issue traced from the referred sources are presented.

Yaling Xun et al. [1] proposed an ultra-metric tree for mining frequent item set ultra-metric provided four advantages of FP and Apriori like partitioning a database in a minimizing input-output and compressed storage. FiDoop - HD is an algorithm designed to overcome the problems like fault tolerance, automatic parallelization, load balancing, and distribution on the large cluster.

B. Al-Maqeleh et.al. [2] had proposed issue of such a process is that the explanation of patterns has to be accomplished merely on frequent item sets. An effective algorithm was introduced to participate in confidence determining throughout of the process of item sets of frequent mining and significantly improves the performance of mining association rules by deducting the search space. The experimental outcome shows that thee effectively of the proposed algorithm in decreasing the number of revealed processes associating with the Apriori algorithm.

Y. Jia et.al. [3] had introduced an enhanced procedure depended on a collaboration of aggressive item-sets counting and data division. The suggested procedure has enhanced the two main difficulties which are meet by a typical Apriori algorithm. First difficulty was to the repeatedly scanning of a transactional database. Second difficulty was to the creation of a huge number of candidate sets. In the of data division, the transactional database was distributed into the n number of parts that did not interconnect with each and every other. First scan task, all the frequent item sets of each division are mined this is called local frequent sets. The second scan, the entire database was scanned again and acquired support degree of all candidate item sets, then determining the global or all frequent item sets. After the task of data division, collaboration item sets computing were used to resolve the candidate items sets prior to scanning database every time. So that the entire procedure requires two times the whole database scan.

R. Chang et al. [4] introduced an improve Apriori algorithm in the level L2 was instantly produced from one time scan across the database without generating candidate sets L1, C1, C2. Apriori improves the algorithm operate efficient horizontal data representation and hash table. Improve Apriori algorithm also enhanced the approach of storage to save space and time. The experimental outcome of Apriori improves the algorithm was advanced as related to fp -growth and Apriori.

Moens et al. [5] proposed two different technique for storing frequent item set in MapReduce structure. First technique DistEclat was the scattered type of only Eclat technique, which elevates the speed which allocating the search space between mappers. Second technique BigFIM usages Apriori and Eclat depended on technique, which is planned with databases that convenient in memory space for removing frequent item sets. The benefit of Dist-Eclat and BigFIM was that it to be responsible for scalability and speed. As compare to BigFIM, Dist-Eclat provided less speed and scalability. Big FIM solved the issue of Dist-Eclat essentially, storing of sub trees needs to the whole database into main memory management, and the whole data set requires expected contact to the number of mappers. BigFIM was a combination of all methods which usage algorithm of Apriori algorithm for creating k-FIs. The algorithm of the Eclat was applicable for to detect frequent item sets. Another item set of the candidate item sets did not fit in memory space of greater depth is the restriction of using Apriori algorithm for creating k-frequent item sets in the algorithm of a BigFIM algorithm and also their speed was very slow down.

Zhou et al. [6] Data partition was a significant task in distributed and parallel computing, when the data set is deals with huge scope, and then it converts in more essential. Here proposed the algorithm of balanced parallel FP-Growth, that progresses algorithm of the original parallel frequent pattern that balancing the load of parallel FP-Growth phase. Experimental outcome presented that the strategy of balanced grouping was comparatively decent grouping strategy. In another stage, to elaborate the

exactness of the strategy of balanced grouping earning more features in deliberation.

Riondato et al. [7] proposed (PARMA algorithm) Parallel Randomized Algorithm was found a set of frequent item sets in the minimum amount of time using sampling method. Parallel Randomized Algorithm store association rules and frequent patterns from accurate data. Outcome of the stored frequent item sets was approximate, which was closer to the initial outcome. It founded that the sampling list using the algorithm of the k-means clustering algorithm. The sample list is similar to clusters. The benefit of PARMA algorithm is that it reduces data repetition, data replication and algorithm execution is also faster.

Liao et al. [8], proposed an MR Pre Post algorithm which was depended on MapReduce structure. MRPrePost was an enhanced design of Pre Post. Performance of the Pre Post algorithm was upgraded the prefix pattern. On the basis of PrePost, the algorithm of the MR Pre Post was suitable for association rules of huge mining data. In case of the algorithm of MRPrePost was higher to Pre Post and PFP. The scalability, stability of the algorithm MRPrePost was better than PrePost and PFP. The mining outcome of MR Pre Post was approximately nearer to the original outcome.

III. PROPOSED SYSTEM:

A. SYSTEM OVERVIEW:

The following figure1 shows proposed system architecture i.e. Fidoop-HD in Hadoop platform, in which Data Owner upload video and user download this video in minimal amount of time

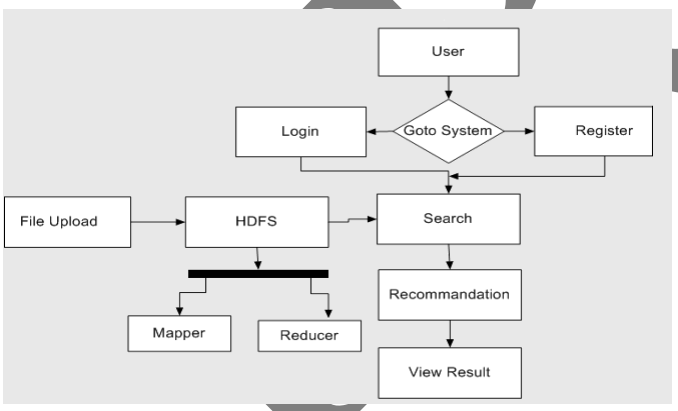

Fig. 1. System Architecture

In Proposed System a new data partitioning method to well balance computing load among the cluster nodes; we develop FiDoop - FIM, an extension of FiDoop, to meet the needs of high dimensional data processing. We have addressed the data-placement issue in heterogeneous Hadoop clusters, where data are placed across nodes in a way that each node has a balanced data processing load. Our data placement scheme is conducive to balancing the amount of data stored in each heterogeneous node to achieve improved data-processing performance. We will integrate FiDoop with the data-placement mechanism on heterogeneous clusters. One of the goals is to investigate the impact of heterogeneous data placement strategy on Hadoop-based parallel mining of frequent item sets.

PSEUDO CODE FOR PROPOSED SYSTEM:

ALGORITHM 1 RECOMMENDED ITEM SET:

Input: k file , k file($2 \le k \le M$) is used to store the frequent k-itemsets generated in the second MapReduce;

Output: Recommended itemset

Pseudocode:

1. Call for function ParallelCounting(key k, values k- file)
2. for all (k is from M to 2) do
3. for all (k-itemset in k- file) do
4. call for function decompose(k-itemset, k-1, (k-1) itemsets); /*Each k-itemset is only decomposed into (k-1)-itemsets */
5. (k-1)-file ← the decomposed (k-1)-itemsets union the original (k-1)-itemsets in (k-1)-file;
6. for all (t-itemset in (k-1)-file) do,
7. Call for function Download itemset(t-itemset);
8. end for
9. end for
10. end for
11. end function

ALGORITHM 2 FUNCTION PARALLEL COUNTING:

Input: minsupport, DBi;

Output: 1-itemsets;

Pseudocode:

1. function MAP(key offset, values DBi)
2. /*T is the transaction in DBi */
3. for all T do
4. items split each T;
5. for all item in items do
6. output(item, 1);
7. end for
8. end for
9. end function
10. reduce input: (item,1)
11. function REDUCE(key item, values 1)
12. initialize sum equal to 0
13. for all item do
14. increment the sum value
15. end for
16. output(1-itemset, sum); /*item is stored as 1-itemset_n
17. if sum ≥ minsupport then
18. F – list ← the (1-itemset, sum) /*F-list is a CacheFile storing */
19. end if
20. end function

ALGORITHM 3 FUNCTION DECOMPOSE:

Input: to be decomposed string s;

Output: the decomposed results l;
Pseudocode:

```
1. function DECOMPOSE(s, l, de-result)
2. /*s is the string to be decomposed, l is the length of the itemset required to be decomposed, de-result stores the results.*/
3. for all ( i is from l to s.length) do
4. decompose(s, i, result, resultend);
5. de-result   i+resultend;
6. end for
7. end function
8. function DECOMPOSE(s, m, result, resultend)
9. if (m == 0) then
10. resultend.addAll(result); //resultend is a list storing all the i-itemset
11. return;
12. end if
13. if (s is not null) then
14. result.add(s[0]+null);
15. for all (j is from the second value to the last value of s) do
16. s1[j-1]   s[j];
17. end for
18. decompose(s1, m - 1, result, resultend); //when selecting the _rst item
19. result.remove(result.size() - 1);
20. desompose(s1, m, result, resultend); //when the first item is not selected
21. end if
22. end function
```

ALGORITHM 4 FUNCTION DOWNLOAD ITEMSET:
Input: t-itemset;
Output: download itemset
Pseudocode:

```
1. apply Mapper
2. if (uservalue.equal(low)) do,
3. foreach (read(Bytes i)) do,
4. compress(bytes i);
5. return bytes;
6. end for
7. else
8. return Bytes;
9. end if
10. apply Reducer
11. write(Bytes B);
```

B. ALGORITHM STEPS:

- User want to download video. User has to make the login into his account. After login the user sends request to admin.
- Admin uploads the requested video on DFS location.
- Now the user can search the video in the search engine and the user will get only the requested video on his system.
- Once the user clicks on the video link he will be able to download the video within seconds and the recommendation section of Fidoop will contain all the previously uploaded file.
- The log file will contain all the records of files that the user has downloaded using Fidoop.
- Reducer file stores the record of how many times the user has downloaded a file.
- In prior techniques such as pfp, Fidoop and Fidoop-HD the execution time was considerably increasing when the number of records increased, compare to which in our proposed system FIM-Fidoop based on FIM algorithm highly reduces the execution speed of downloading as the number of records increases in size.

IV. EXPERIMENTAL RESULT:

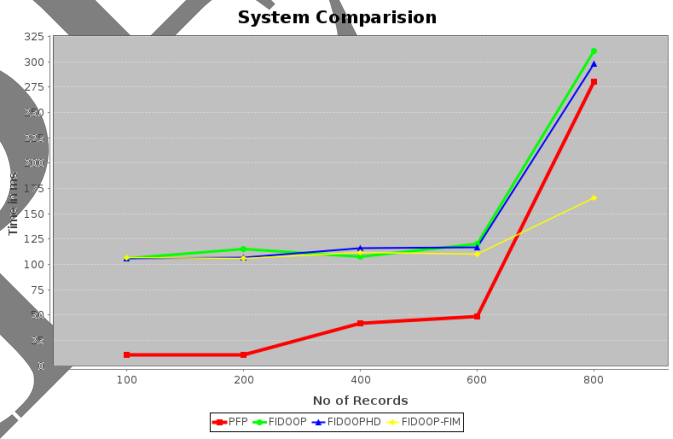

Fig. 2. Show the number of records comparison in proposed system and existing system. It can be seen that the proposed system takes less time as compared to the existing system.

V. CONCLUSION AND FUTURE SCOPE:

MapReduce programming system is concern for current mining algorithm for frequent item sets from the database and solves the scalability and load balancing issue. This paper gives the overview of the algorithms designed for parallel mining of frequent item sets. The FIM and MapReduce algorithm were used for mining frequent item sets. The disadvantage of Frequent Pattern growth though deceits within the insuperable to create in-memory Frequent Patterns trees to put up huge scalable databases. This problem comes to be numerous noticeable comes after from two-dimensional databases or large database. To solve these issue, FiDoop system developed a parallel frequent item set mining algorithm. FiDoop includes Frequent Pattern growth and FIUT algorithm. The FIUT reaches flattened storage. FiDoop executes multiple

MapReduce tasks of which final MapReduce task is important. In that the mapper independently decomposes item sets and reducer built the ultra-metric trees. As compare to pfp, Fidoop and Fidoop-HD an extension of Fidoop the proposed system is an advance technique of Fidoop-FIM which uses FIM algorithm to highly reduce the execution speed of downloading as the number of records increases in size.

In future work we invent the algorithm which is very useful to process the big data using Fidoop-FIM and it also improves the efficiency existing hadoop. By considering users review as an important aspect, working of implemented system can be improved.

ACKNOWLEDGMENT:

This paper would not have been come into reality without the able guidance, support and wishes of all those who stand by me in the development. I wish to give my special thanks to my guide, Prof. Ekta Ukey, for her timely advice and guidance.

REFERENCES:

1) Yaling Xun, Jifu Zhang, and Xiao Qin, "*FiDoop: Parallel Mining of Frequent Itemsets Using MapReduce*", 2 IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS 2015.
2) Basheer Mohamad Al-Maqaleh and Saleem Khalid Shaab," *An Efficient Algorithm for Mining Association Rules using Confident Frequent Itemsets,"* 2012 Third International Conference on Advanced Computing & Communication Technologies.
3) Yubo Jia, Guanghu Xia, Hongdan Fan, Qian Zhang and Xu Li, "*An Improved Apriori Algorithm Based on Association Analysis,"* ICNDC 2012, 3rd IEEE International Conference, pp208-211.
4) Rui Chang and Zhiyi Liu , " *An Improved Apriori Algorithm*," ICEOE 2011, IEEE International Conference, vol. 1, pp v1- 476 -v1-478.
5) Moens S , Aksehirli E , Goethals B , —*Frequent Itemset Mining for Big Data,||* Big Data, 2013 IEEE International Conference on , vol., no., pp.111,118, 6-9 Oct. 2013 DOI: 10.1109 / Big Data. 2013.6691742.
6) L. Zhou, Z. Zhong, J. Chang, J. Li, J. Huang, and S. Feng. —*Balanced parallel FP Growth with MapReduce*||. In Proc.YC-ICT, pages 243–246, 2010.
7) Riondato, J. A. DeBrabant, R. Fonseca, and E. Upfal. —PARMA: *a parallel randomized algorithm for approximate association rules mining in MapReduce||.* In Proc. CIKM, pages 85–94. ACM, 2012.
8) Jinggui Liao, Yuelong Zhao, and Saiqin Long,—*MRPrePost- A Parallel algorithm adapted for mining big data,|| IEEE Workshop on Electronics,* Computer and Applications, 2014.