Efficient Frequent Itemset Mining On Bigdata Using FIU-tree

Hanumanthu T C[1], Arun Kumar[2]

[1]*Student, Dept. of CSE, MVJCE, Bangalore, Karnataka, India*
[2]*Assistant professor, Dept. of CSE, MVJCE, Bangalore, Karnataka, India*

---***---

Abstract—*Existing parallel mining algorithms for frequent itemsets lack a mechanism that enables automatic parallelization, load balancing, data distribution, and fault tolerance on large clusters. As a solution to this problem, we design a parallel frequent itemsets mining algorithm called FiDoop using the MapReduce programming model. To achieve compressed storage and avoid building conditional pattern bases, FiDoop incorporates the frequent items ultrametric tree, rather than conventional FP trees. In FiDoop, three MapReduce jobs are implemented to complete the mining task. In the crucial third MapReduce job, the mappers independently decompose itemsets, the reducers perform combination operations by constructing small ultrametric trees, and the actual mining of these trees separately. We implement FiDoop on our in-house Hadoop cluster. We show that FiDoop on the cluster is sensitive to data distribution and dimensions, because itemsets with different lengths have different decomposition and construction costs. To improve FiDoop's performance, we develop a workload balance metric to measure load balance across the cluster's computing nodes. We develop FiDoop-HD, an extension of FiDoop, to speed up the mining performance for high-dimensional data analysis. Extensive experiments using real-world celestial spectral data demonstrate that our proposed solution is efficient and scalable.*

Keywords—Frequent itemsets, frequent items ultrametric tree (FIU-tree), Hadoop cluster, load balance, MapReduce.

I. introduction

Parallel Frequent Itemset mining is looking for sequence of actions and load balancing of dataset. Creating Hadoop cluster is especially for storage and analyzing data. Through frequent Itemset mining extracting knowledge from data. Example of this technique is Market Basket Algorithm. It also affect on load balancing. It helps to increase the speed of performance. This parallel Frequent Itemset mining is done using map reduce programming model. Partitioning of data in dataset through algorithm making data more efficient. This data partitioning is carried out on Hadoop clusters. Data partitioning necessary for scalability and high efficiency in cluster. In Frequent Itemsets Mining data partition affects to computing nodes and the traffic in network. Data partition may be spread over multiple nodes, and users at the node can perform local transactions on the partition. This increases performance for sites that have regular transactions involving certain views of data, whilst maintaining availability and security. By using Fidoop-DP concept, performance of parallel Frequent Itemset Mining on Hadoop clusters increases. Fidoop-DP is voronoi diagram. It is conceptualized on data partition strategy.

II. literature survey

Distinctive techniques have been put for slim the writing review to address the issue, when datasets in current information mining applications turn out to be too much vast, consecutive FIM calculations running on a solitary machine experience the ill effects of execution disintegration. The going with fragment shows a segment of the techniques used for this reason. All the more vitally, the current parallel algorithms do not have an instrument that empowers programmed parallelization, load adjusting, information dispersion, and adaptation to non-critical failure on huge figuring bunches.

[1] This Paper proposes how frequent itemset mining finds much of the time happening itemsets in value-based information. This is connected to assorted issues, for example, decision backing, specific promoting, money related gauge and medicinal analysis. The cloud, calculation as a utility administration, permits us to crunch expansive mining issues. There are various calculations for doing visit itemset mining, yet none are out-of-the-crate suited for the cloud, requiring vast information structures to be synchronized over the system. The greatest calculations meant for liability visit itemset mining are the famous FP-development (Frequent Patterns development).

[2] This paper proposes MapReduce is a programming model for handling and producing extensive information sets.

We fabricated a framework around this programming model in 2003 to disentangle development of the upset list for taking care of hunts at Google.com. From that point forward, more than 10,000 particular projects have been actualized utilizing MapReduce at Google, including calculations for extensive scale diagram handling, content preparing, machine learning, and factual machine interpretation. The

Hadoop open source execution of MapReduce has been utilized widely outside of Google by various associations.

[3] This paper proposes Efficient get ready of neighbor joins using MapReduce k nearest k closest neighbor jo.in (kNN join), intended to discover k closest neighbors from a dataset S for each item in another dataset R, is a primitive operation generally received by numerous information mining applications. To some things up, the mappers bunch objects into gatherings; the reducers perform the kNN join on every gathering of items independently.

We outline a viable mapping instrument that endeavors pruning rules for separation sifting, and thus diminishes both the rearranging and computational expenses. To decrease the rearranging cost, we propose two rough calculations to minimize the quantity of re productions. Broad investigations on our in-house group exhibit that our proposed strategies are proficient, strong and adaptable.

[4] This paper proposes the interrelation examination of grand spectra data using constrained consistent pattern trees Association principle mining, in which creating continuous examples is a key stride, is a successful method for recognizing inalienable and obscure interrelationships between qualities of divine spectra information and its physicochemical properties. In this study, we first make utilization of the main request predicate rationale to speak to learning got from heavenly spectra information. Next, we propose an idea of obliged regular example trees (CFP) alongside a calculation used to develop CFPs, planning to enhance the productivity and relevance of affiliation tenet mining.

[5] This paper proposes a heap adjusted appropriated parallel mining algorithms. Because of the exponential development in overall data, organizations need to manage a regularly developing measure of computerized data. A standout amongst the most imperative difficulties for information mining is rapidly and effectively finding the relationship among information. The Apriori calculation has been the most prevalent procedure in finding incessant examples. Be that as it may, while applying this strategy, a database must be checked commonly to ascertain the tallies of an immense number of applicant itemsets. Parallel and appropriated registering is a successful technique for quickening the mining procedure.

[6] This paper proposes DH-TRIE incessant example mining on Hadoop utilizing JPA. The FP-growth is an understood relentless case's estimation in data mining when working with high-dimensional, vast scale information sets. It is otherwise called awesome multifaceted nature on memory for the recursively preparing. When all is said in done, FP growth can't deal with substantial scale information set unless isolating an entire information set into little squares. Taking into account Hadoop, the open distributed computing show, a disseminated DH-TRIE regular example calculation utilizing JPA is proposed, which tackled the three issues (globalization, arbitrary compose and length). The calculation is indicated great adaptability and versatility by correlations with mahout venture. By connected to a virtualization stage Vega Cloud, the calculation will be utilized as a part of far-extending circumstances.

III. SYSTEM DESIGN

A meaningful representation of the system to be developed in any research work is known as design. The interaction among the modules requires high I/O and multiple threads. The main consideration is to make the model and the system more compatible so that both the entity proves to be efficient. The process by which this task is carried out combines the benchmark result based o case studies which are a set of input to this research.

Fidoop architectural overview is been focused and demonstrated in the following section; figure 1. projects the architected diagram. The system architecture consists of a upload process, preprocessing job, generation of frequent itemsets using FP-growth and FIUT technique for development of system protocol design and analysis. This system is featured to collect the data from the independent sources under a privileged authenticated status. The graph generated shows the time taken by the two algorithms to do the frequent itemset process based on the support value and number of records.

The graph projects the overall status on developing and designing the system requirement as per the resource availability. In our proposed system we have discussed about online retail. Each time a stipulated system is generated and thus its acquired results are analyzed and added.

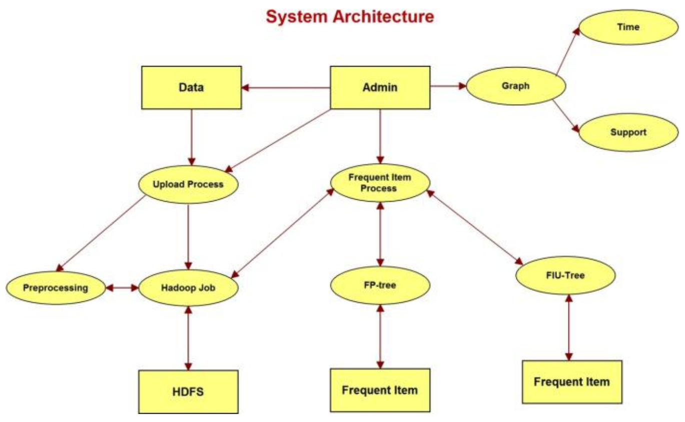


Figure 1: Architecture of Proposed System

A. **FIUT:**

The FIUT approach adopts the FIU-tree to enhance the efficiency of mining frequent itemsets. FIU-tree is a tree structure constructed as follows.

1) After the root is labeled as null, an itemset p1, p2, . . . , pm of frequent items is inserted as a path connected by edges (p1, p2), (p2, p3), . . . , (pm−1, pm) without repeating nodes, beginning with child p1 of the root and ending with leaf pm in the tree.

2) An FIU-tree is constructed by inserting all itemsets as its paths, each itemset contains the same number of frequent items. Thus, all of the FIU-tree leaves are identical height.

3) Each leaf in the FIU-tree is composed of two fields: named item-name and count. The count of an item-name is the number of transactions containing the itemset that is the sequence in a path ending with the item name. Non leaf nodes in the FIU-tree contains two fields: named item-name and node-link. A node-link is a pointer linking to child nodes in the FIU-tree.

The FIUT algorithm consists of two key phases. The first phase involves two rounds of scanning a database. The first scan generates frequent one-itemsets by computing the support of all items, whereas the second scan results in k-itemsets by pruning all infrequent items in each transaction record. Note that, k denotes the number of frequent items in a transaction. In phase two, a k-FIU-tree is repeatedly constructed by decomposing each h-itemset into k-itemsets, where $k + 1 \leq h \leq M$ (M is the maximal value of k), and unioning original k-itemsets. Then, phase two starts mining all frequent k-itemsets based on the leaves of k-FIU-tree without recursively traversing the tree. Compared with the FP-growth method, FIUT significantly reduces the computing time and storage space by averting overhead of recursively searching and traversing conditional FP trees..

B. **MapReduce Framework:**

MapReduce is a promising parallel and scalable programming model for data-intensive applications and scientific analysis. A MapReduce program expresses a large distributed computation as a sequence of parallel operations on datasets of key/value pairs. A MapReduce computation has two phases, namely, the Map and Reduce phases. The Map phase splits the input data into a large number of fragments, which are evenly distributed to Map tasks across the nodes of a cluster 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 keyvalue pairs. Both input pairs of Map and the output pairs of Reduce are managed by an underlying distributed file system. MapReduce greatly improves programmability by offering automatic data management, highly scalable, and transparent fault-tolerant processing. Also, MapReduce is running on clusters of cheap commodity servers—an increasingly attractive alternative to expensive computing platforms. Thanks to the aforementioned advantages, MapReduce has been widely adopted by companies like Google, Yahoo, Microsoft, and Facebook.

Hadoop—one of the most popular MapReduce implementations—is running on clusters where Hadoop distributed file system (HDFS) stores data to provide high aggregate I/O bandwidth. At the heart of HDFS is a single Name Node—a master server that manages the file system namespace and regulates access to files. The Hadoop runtime system establishes two processes called Job Tracker and Task Tracker. Job Tracker is responsible for assigning and scheduling tasks; each Task Tracker handles Map or Reduce

C. **Background Subtraction:**

The idea of background subtraction method is to initialize a background firstly, and then the current frame is subtracted with reference frame to detect moving object. This method is simple and easy to realize, and accurately extracts the characteristics of target data .The output image is the binary image. Morphological filtering is applied to that image and it will perform the operation like opening, closing, sharpening the edges and it will also remove the noise from that frame. The function of the morphological filtering is the removal of small regions created by noise; fill up unnecessary gaps, smoothing boundaries, extracting edges. It will give pixel level operations.

IV. implementation Details

Now, we discuss the implementation details of FiDoop. We pay particular attention to the last MapReduce job in FiDoop, because the last job is computationally expensive. We show how to optimize the performance of the third MapReduce job.

A. Load Balance:

The *decompose*() function of the third MapReduce job accomplishes the decomposition process. If the length of an itemset is *m*, the time complexity of decomposing the itemset is $O(2m)$. Thus, the decomposition cost is exponentially proportional to the itemset's length. In other words, when the itemset length is going up, the decomposition overhead will dramatically enlarged. The data skewness problem is mainly induced by the decomposition operation, which in turn has a significant performance impact on FiDoop. The first step toward balancing load among data nodes of a Hadoop cluster is to quantitatively measure the total computing load of processing local itemsets. We achieve this first step by developing a workload-balance metric to quantify load balance among the data nodes.

B. High-Dimensional Optimization:

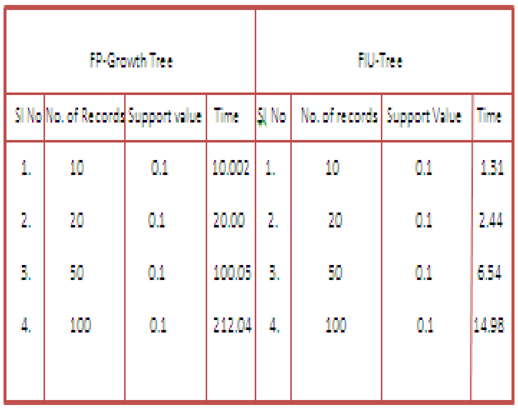
The aforementioned analysis confirms that if the length of itemsets to be decomposed is large, the decomposition cost will exponentially increase. In this section, we conduct experiments to investigate the impact of dimensionality on FiDoop. We also compare FiDoop with a popular solution parallelization of FP-growth (Pfp) . Section VI presents an optimization algorithm called FiDoop-HD for high-dimensional data processing.

When it comes to mining frequent itemsets, varying dimensionality leads to a wide range of itemset lengths. Our algorithm needs to decompose each itemset generated by pruning infrequent items for each transaction. We made use of the series of D1000W, which are described in detail in Section VII (see Synthetic Dataset). In the group of experiments, the number of transactions is 10 000 and the average transaction size is anywhere between 10 and 50.

V. Performance Analysis

The efficiency of the system can be analyzed in terms of time taken by the FP-Tree and FIU-Tree algorithms in generating the Frequent Itemsets. We compare the performance of the system with the Fidoop system.

In Table 5.1 shows the time taken to create the Frequent Itemsets of different transactions size. From the Table, it is clear that the amount of time taken to generate frequent itemsetsby FP-Tree algorithm is around 1.5 times slower than the FIU-Treefor the same input.



FP-Growth Tree				FIU-Tree			
Sl No	No. of Records	Support value	Time	Sl No	No. of records	Support Value	Time
1.	10	0.1	10.002	1.	10	0.1	1.31
2.	20	0.1	20.00	2.	20	0.1	2.44
3.	50	0.1	100.05	3.	50	0.1	6.54
4.	100	0.1	212.04	4.	100	0.1	14.98

Table 5.1 Performance analysis with varied Record size

Minimum support plays an important role in mining frequent itemsets. We increase minimum support thresholds from 0.0001% to 0.0003% with an increment of 0.00005%. Figure 5.2 shows evaluating the impact of minimum support on Pfp and our proposed algorithms containing three MapReduce jobs using both celestial spectral and synthetic datasets.

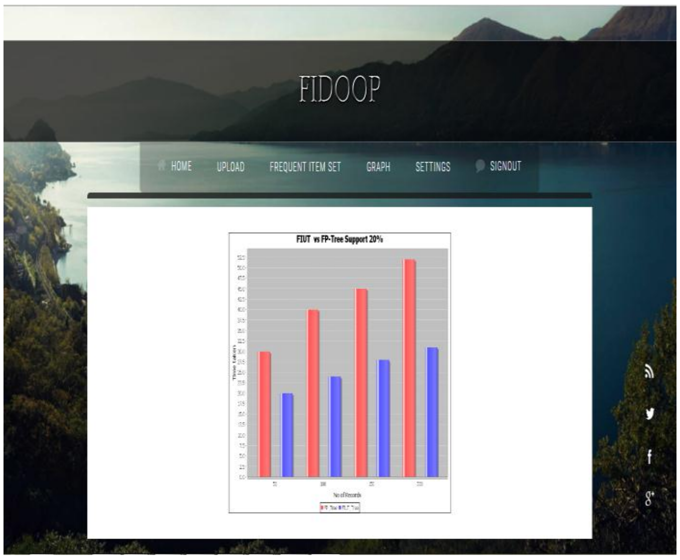


Fig. 5.2: Frequent Itemset Generation Time

Fig. 5.3 shows the impact of workload balance metric on running time measured in the unit of 1000 s.

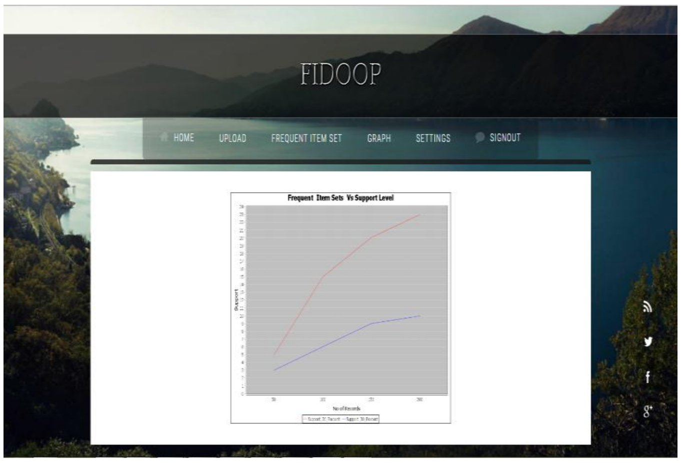


Fig.5.3 Frequent Itemsets vs Support Level

VI. Conclusion

The current frequent itemset mining algorithm is facing big challenges in load balancing, efficiency and scalability, to overcome these challenges our proposed system uses MapReduce techniques and Hadoop. Our proposed algorithm performs three MapReduce jobs to get Frequent Itemsets. The data is collected from UCI datasets which are preprocessed and uploaded into Hadoop .When frequent itemsets mining algorithm is invoked, the datasets are fetched from Hadoop and mining algorithms are processed on data and produced frequent itemsets. Fidoop system has been dedicated to produce an accurate data mining results under Hadoop single node cluster environment,

The system achieves high efficiency gain for providing static information resources for dynamic and critical data under big data mining. Results are detailed and discussed in previous chapters with overall system design and analysis. This System is developed as a web based application in MVC architecture on Java platform. For comparison FP-Tree and FIU-Tree techniques are used.

The present dataset used belongs to online retail, in future we can use different datasets from different verticals, also we can develop mobile app to get results in our hand. This system in future can be enhanced with a diplomatic sentiment analysis and redefine process of computation under big data environment.

References

[1] Rizwana Kowsar M.S and Somesekhar T "Data Heirarchy Mining under MapReduce Techniques for Frequent Itemsets" Proceedings of International Conference on Computers, Communications, Control and Applied Science ThinkIT-2016,All copyrights @ IRD Journals 2016 vol. 10, pp.148-153.

[2] Rizwana Kowsar M.S and Somesekhar T "Dataset based MapReduce technique under Chronic Mining and Confidence analysis" 1st International Conference on Internet of Things (ICIOT -2016) conducted by the Department of Computer Science and Engineering APS College of Engineering.

[3] W. Lu, Y. Shen, S. Chen, and B. C. Ooi, "Efficient processing of k nearest neighbor joins using MapReduce," Proc. VLDB Endow., vol. 5, no. 10, pp. 1016–1027, 2012.

[4] J. Zhang, X. Zhao, S. Zhang, S. Yin, and X. Qin, "Interrelation anal- ysis of celestial spectra data using constrained frequent pattern trees," Knowl.-Based Syst., vol. 41, pp. 77–88, Mar. 2013.

[5] K. Yu and J. Zhou, "Parallel TID-based frequent pattern mining algo- rithm on a PC cluster and grid computing system," Expert Syst. Appl., vol. 37, no. 3, pp. 2486–2494, 2010.

[6] "Distributed Algorithm for Frequent Pattern Mining using HadoopMap Reduce Framework"

Suhasini A. Itkar1, Uday V. Kulkarni2 1 PES Modern college of Engineering, Pune, India.DOI: 02.AETACS.2013.4.123 © Association of Computer Electronics and Electrical Engineers, 2013.

[7] K.-M. Yu, J. Zhou, T.-P.Hong, and J.-L. Zhou, "A load-balanced dis- tributed parallel mining algorithm," Expert Syst. Appl., vol. 37, no. 3, pp. 2459–2464, 2010.

[8] K. W. Lin, P.-L. Chen, and W.-L. Chang, "A novel frequent pattern mining algorithm for very large databases in cloud computing environ- ments," in Proc. IEEE Int. Conf. Granular Comput. (GrC), Kaohsiung, Taiwan, 2011, pp. 399–403.

[9] L. Yang, Z. Shi, L. D. Xu, F. Liang, and I. Kirsh, "DH-TRIE frequent pattern mining on Hadoop using JPA," in Proc. IEEE Int. Conf. Granular Comput. (GrC), Kaohsiung, Taiwan, 2011, pp. 875–878.

[10] "Mining Distributed Frequent Itemset with Hadoop" Ms. Poonam Modgi, PG student, Parul Institute of Technology, GTU. Prof. Dinesh Vaghela, Parul Institute of Technology, GTU.Poonam Modgi et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3), 2014, pg. 3093 – 3097.