ISSN:2348-2079

Volume-6 Issue-2

International Journal of Intellectual Advancements and Research in Engineering Computations

# Distributed Pattern Discovery using Load Management with Resource and Data Constraints under Clouds

**Ninumol C.P., MTech (CSE) ,**
**Bhavya K Bharathan, MTech ., Assistant Professor,**
Cochin College of Engineering and Technology, Valiyaparambu, Edayur P.O.,Valanchery, Malappuram District, Kerala, India
**Mail ID: ninumolcp@gmail.com**

**Abstract**

Huge volume of data values are collected and managed under the big data models. The big data mining operations are performed with the support of the storage and computational resources from the clouds. The big data analytics are carried out using the Hadoop environment with parallel and distributed computing support. The homogeneous and heterogeneous computing models are managed with the Hadoop environment. The data and tasks are partitioned and processed as small elements in the MapReduce framework. The mapreduce framework is adapted to manage the data and task as tiny elements.

The association rule mining methods are applied to detect the frequent patterns in the data values. The data values are equally divided and processed under the parallel frequent pattern mining techniques. The rule mining with load management operations are carried out with the Data Partitioning in Frequent Itemset Mining on Hadoop Clusters (FiDoop-DP). The transaction relationships are analyzed in the Voronoi diagram based data partitioning scheme. The similarity metric and Locality Sensitive Hashing (LSH) technique are applied to manage the redundant data values. The Parallel Frequent Pattern Growth algorithm is employed to discover the rules using the cloud resources.

The data pattern discovery operations are designed with resource and data constraints. The dynamic data partitioning and transmission over the Hadoop clusters is supported in the parallel rule mining process. The dynamic resource level based computational nodes are used to construct the heterogeneous Hadoop clusters. The load management in data partitioning operations are carried out with data and resource constraints. The FiDoop-DP scheme is enhanced to handle the data placement with load management under the Hadoop Distributed File System (HDFS) in heterogeneous nodes. The computational and communication loads are minimized with the parallel frequent pattern mining process.

**Index Terms:** Big data analysis, Hadoop clusters, Parallel rule mining, Distributed File Systems (DFS) and Locality Sensitive Hashing (LSH).

## 1. Introduction

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 itemsetsw hose 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.

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). 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 assigning and scheduling tasks; each TaskTracker handles mappers or reducers assigned by JobTracker. When Hadoop exhibits an overwhelming development momentum, a new MapReduce programming model Spark attracts researchers' attention. The maina bstraction 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 FiDoop-DP on Hadoop clusters; in a future study, we plan to extend FiDoop-DP to Spark to gain further performance improvement.

## 2. Related Work

### 2.1. Mining of Frequent Itemsets

The Apriori algorithm is a classic way of mining frequent itemsets in a database. A variety of Apriori-like algorithms aim to shorten database scanning time by reducing candidate itemsets. For example, Park et al. proposed the direct hashing and pruning algorithm to control the number of candidate two-itemsets and prune the database size using a hash technique. In the inverted hashing and pruning algorithm, every k-itemset within each transaction is hashed into a hash table. Berzal et al. designed the tree-based association rule algorithm, which employs an effective data-tree structure to store all itemsets to reduce the time required for scanning databases.

To improve the performance of Apriori-like algorithms, Han et al. proposed a novel approach called FP-growth to avoid generating an excessive number of candidate itemsets. The main idea of FP-growth is projecting database into a compact data structure, and then using the divide-and-conquer method to extract frequent itemsets. The main bottlenecks of FP-growth are: 1) the construction of a large number of conditional FP trees residing in the main memory and 2) the recursive traverse of FP trees. To address this problem, Tsay et al. [8] proposed a new method called FIUT, which relies on frequent items ultrametric trees to avoid recursively traversing FP trees. Zhang et al. [10] proposed a concept of constrained frequent pattern trees to substantially improve the efficiency of mining association rules.

### 2.2. Parallel Mining of Frequent Itemsets

Parallel frequent itemsets mining algorithms based on Apriori can be classified into two camps, namely, count distribution, fast parallel mining and data distribution. In the count distribution camp, each processor of a parallel system calculates the local support counts of all candidate itemsets. All processors compute the total support counts of the candidates by exchanging the local support counts. In the data distribution camp, each processor only keeps the support counts of a subset of all candidates. Each processor is responsible for sending its local database partition to all the other processors to compute support counts. DD has higher communication overhead than CD, because shipping transaction data demands more communication bandwidth than sending support counts.

The cascade running mode in existing Apriori-based parallel mining algorithms leads to high communication and synchronization overheads. To reduce time required for scanning databases and exchanging candidate itemsets, FP-growth based parallel algorithms were proposed as a replacement of the Apriori-based parallel algorithms. A few parallel FP-growth-based parallel algorithms were implemented using multithreading on multicore processors. A major disadvantage of these parallel mining algorithms lies in the infeasibility to construct main-memory-based FP trees when databases are very large. This problem becomes pronounced when it comes to massive and multidimensional databases.

## 2.3. Parallel Data Mining on Clusters

Clusters and other shared-nothing multiprocessor systems are scalable computing platforms that address the aforementioned main memory issue raised in parallel mining of large-scale databases. Tang and Turkia used the extended conditional databases and k-prefix search space partitioning to parallelize FIM, and an implementation of the new scheme with FP trees is presented. Yu and Zhou [9] proposed two parallel mining algorithms, Tidset-based parallel FP-tree (TPFP-tree) and balanced Tidset-based parallel FP-tree (BTP-tree). The TPFPtree algorithm uses a transaction identification set to directly select transactions rather than scanning an entire database with a vertical database layout.

To optimize the performance of heterogeneous computing environments, Yu et al. [1] developed the BTP-tree algorithm, which is an extension of the TPFP-tree algorithm. The load balancing scheme adopted in the BTPtree algorithm takes into account the heterogeneous processing power of computing nodes, thereby delivering an effective approach to parallelmining of frequent patterns. Zhou et al. [2] proposed a balanced parallel FP-growth algorithm balanced parallel FP-growth based on the Pfp algorithm.

Recently, increasing attention has been paid to supporting parallel data mining over cloud computing environments and the MapReduce framework [11]. For example, Li et al. proposed the Pfp algorithm on distributed machines. Pfp partitions computation to make each machine execute an independent group of mining tasks, thus, eliminating communication cost. Lin et al. [3] proposed a method for mining frequent itemsets from very large database. It used an efficient data structure for storing and retrieving FP trees and used the disk as the secondary memory. Yang et al. [4] proposed a distributed Distributed Hash-Trie frequent pattern algorithm using java persistence API based on Hadoop. Riondato et al. [5] proposed PARMA which cut down the data-size-dependent part of the cost by using a random sampling approach to FIM to improve the efficiency of mining. Hong et al. [6] proposed an improved FP-growth algorithm in MapReduce for discovering frequent patterns. Choi et al. [7] designed a scheme for applying MapReduce to the FP-growth algorithm to ensure efficient distribution of important resources. Different from the above techniques that make use of FP trees, our FiDoop incorporates FIU trees to achieve compressed storage and avoid building conditional pattern bases.

## 3. Pattern Discovery under Hadoop Clusters

Traditional parallel Frequent Itemset Mining techniques (FIM) are focused on load balancing; data are 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, redundant transaction transmission and itemset-mining tasks are likely to be created by inappropriate data partitioning decisions.

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. We propose a parallel FIM approach called FiDoop-DP using the MapReduce programming model. The key idea of FiDoop-DP is to group highly relevant transactions into a data partition; 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. FiDoop-DP is conducive to speeding up the performance of parallel FIM on clusters.

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. FIM algorithms running on a single machine suffer from performance deterioration due to limited computational and storage resources. 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. In the realm of big data processing, MapReduce has been adopted to develop parallel data mining algorithms, including Frequent Itemset Mining. Hadoop is an open source implementation of the MapReduce programming model. 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 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 FIM built in Hadoop aim at balancing computation load by equally distributing data among nodes. 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 FiDoop-DP, a parallel FIM technique, large dataset is partitioned across a Hadoop cluster's data nodes in a way to improve data locality.

## 4. Problem Statement

The frequent item set mining methods are applied to fetch frequent patterns from the database transactions. The parallel frequent mining techniques divide and process the data set with equal intervals. Inappropriate data partition initiates redundant transaction transmission and item set mining tasks. The Data Partitioning in Frequent Itemset Mining on Hadoop Clusters (FiDoop-DP) is adapted to perform the load balanced rule mining process. The data partitions are created with highly relevant transaction groups. The Voronoi diagram based data partitioning scheme uses the transaction relationships. The partitioning process controls the redundant transactions with similarity metric and Locality Sensitive Hashing (LSH) technique. The Parallel Frequent Pattern Growth algorithm is employed to discover the frequent item sets. The following problems are identified from the current Hadoop based rule mining techniques. Computational resource status is not considered in the data partitioning process. Data distribution and rule mining methods are not tuned for heterogeneous environment. Data placement loads are not focused for Hadoop Distributed File Systems (HDFS). Energy management models are not focused in the rule mining process.

## 5. Distributed Pattern Discovery using Load Management under Clouds

The association rule mining methods are used to fetch frequent item sets from the database transactions. The distributed database mining operations can be performed with the support of the cloud resources. The Hadoop clusters are formed with homogeneous and heterogeneous computation nodes. The resources from Hadoop clusters can be used for the parallel rule mining process. The transactional data can be partitioned to a set of groups. The data partitions are passed to the computational nodes in the Hadoop cluster. The mining operations are carried out on the partitioned data values. The data partitions are managed under the Hadoop Distributed File System (HDFS). The data partitioning is carried out with load balancing concepts. The Frequent Itemset Mining on Hadoop Clusters (FiDoop-DP) scheme is used in the mining process. The Parallel Frequent Pattern Growth (PFPG) algorithm is used for the rule mining process. Locality Sensitive Hashing (LSH) is used for the data partitioning process.

Parallel rule mining process is build to support dynamic data partitioning and distribution over the heterogeneous Hadoop clusters. The heterogeneous Hadoop clusters are build on computational nodes with different resource levels. The data aware partitioning process is carried out with the computational resource and load level details. The load levels are dynamically estimated from the Hadoop cluster environment. The FiDoop-DP scheme is upgraded to handle the load balanced data placement under the HDFS in heterogeneous nodes. The data placement operations are carried out to transfer the partitioned data values to the computational nodes in the Hadoop clusters. The data placement is dynamically planned and executed with reference to the current load and resource levels. The parallel frequent Item set mining process is improved with energy efficiency features. The energy management is handled with minimizing the computational load and data distribution loads.

## 5.1. Data Partitioning

FIM is a multi-stage parallel process, where redundant transactions transmission and redundant mining tasks occur in the second MapReduce job. Recall that it is a grand challenge to avoid these downsides by using traditional grouping strategies and default partitioning function. And transferring redundant transactions is a main reason behind high network load and redundant mining cost. To solve this problem, we propose to partition transactions by considering correlations among transactions and items prior to the parallel mining process. That is, transactions with a great similarity are partitioned into one partition in order to prevent the transactions from being repeatedly transmitted to remote nodes. We adopt the Voronoi diagram-based data partitioning technique is conducive to maintaining data proximity, especially for multi dimensional data. Therefore, when the second MapReduce job is launched, a new Voronoi diagram based data partitioning strategy is deployed to minimize unnecessary redundant transaction transmissions.

Voronoi diagram is a way of dividing a space into a number of regions. A set of points referred to as pivots is specified beforehand. For each pivot, there is a corresponding region consisting of all objects closer to it han to the other pivots. The regions are called Voronoi cells. The idea of Voronoi diagram-based partitioning can be formally described as follows. Voronoi diagram-based partitioning selects k objects as pivots. Then, all objects of D are split into k disjoint partitions, where each object is assigned to the partition with its closest pivot. In this way, the entire data space is split into k cells. Incorporating the characteristic of FIM, we adopt the similarity as the distance metric between transaction and pivot in Voronoi diagram. In addition, Voronoi diagram-based partitioning relies on a way of selecting a set of pivots. Thus, in what follows, we investigate distance measure and pivot-selection strategies, followed by partitioning strategies.

### 5.1.1. Distance Metric

Recall that to optimize FIM, a good partitioning strategy should cluster similar data objects to the same partition Similarity is a metric to quantitatively measure the correlation strength between two objects. To capture the characteristics of transactions, we adopt the Jaccard similarity as a distance metric. Jaccard similarity is a statistic commonly used for comparing the similarity and diversity of sample data objects. A high Jaccard similarity value indicates that twodata sets are very close to each other in terms of distance.

In order to quantify the distance among transactions, we model each transaction in a database as a set. Then, the distance among transactions is measured using the Jaccard similarity among these sets. The Jaccard similarity of two sets A and B is defined as
J(A; B) = |AnB| \ |AUBj|

Obviously, J(A;B) is a number ranging between 0 and 1; it is 0 when the two sets are disjoint, 1 when they are identical, and strictly between 0 and 1 otherwise. That is, the distance between two sets is close when their Jaccard index is closer to 1; if there is a large distance between the two sets, their Jaccard index is closer to 0.

### 5.1.2. K-means Selection of Pivots

Intuitively, selecting pivots directly affects the uniformity coefficient of the remaining objects for voronoi diagrambased partitioning. In particular, we employ the K-meansbased selection strategy choose pivots. And the pivot selecting process is conducted as a data preprocessing phase. K-means is a popular algorithm for clustering analysis in data mining. K means clustering aims to partition n objects into k clusters. That is, given a set of objects ($x_1$; $x_2$; ; $x_n$), where each object is a d-dimensional real vector, k means clustering partitions the n objects into k (k , n) sets C = $C_1$;$C_2$; ;$C_k$, in each object belongs to a cluster with the nearest mean. The clustering results can be applied to partition the data space into Voronoi cells. To reduce the computational cost of k-means, we perform sampling on the transaction database before running the kmeans algorithm. It is worth mentioning that the selection of initial pivots plays a critical role in clustering performance.

### 5.1.3. LSH-based Partitioning

Upon the selection of pivots, we calculate the distances from the rest of the objects to these pivots to determine a partition to which each object belongs. We develop the LSH-based strategy to implement a novel grouping and partitioning process, prior to which MinHash is employed as a foundation for LSH.

Locality sensitive hashing, or LSH, boosts the performance of MinHash by avoiding the comparisons of a large number of element pairs. Unlike MinHash repeatedly evaluating an excessive number of pairs, LSH scans all the transactions once to identify all the pairs that are likely to be similar. We adopt LSH to map transactions in the feature space to a number of buckets in a way that similar transactions are likely to be mapped into the same buckets.

## 6. Conclusion and Future Work

The parallel rule mining methods are applied to discover the frequent item set under the distributed environment. Homogeneous and Heterogeneous clusters are used to provide the storage and computational resources for the distributed rule mining process. The load balanced rule discovery process is carried out using the Data Partitioning in Frequent Itemset Mining on Hadoop Clusters (FiDoop-DP). The FiDoop-DP scheme is improved with load balanced data partitioning, data placement and energy management models. The Data Partitioning in Frequent Itemset Mining on Hadoop Clusters (FiDoop-DP) scheme is adapted to perform parallel rule mining. Data partitioning and data placement operations are tuned for the heterogeneous clusters. Similarity redundant data filtering model minimizes the data transmission load. The system improves the energy efficiency with minimum computational complexity levels. The parallel rule mining scheme can be enhanced to support utility based rule mining and weighted rule mining concepts. The system can be improved to protect sensitive data items in the rule mining process.

## REFERENCES

[1] K.-M. Yu, J. Zhou, T.-P. Hong, and J.-L. Zhou, "A load-balanced distributed parallel mining algorithm," Expert Syst. Appl., vol. 37, no. 3, pp. 2459–2464, 2010.
[2] L. Zhou et al., "Balanced parallel FP-growth with MapReduce," in Proc. IEEE Youth Conf. Inf. Comput. Telecommun. (YC-ICT), Beijing, China, 2010.
[3] K. W. Lin, P.-L. Chen, and W.-L. Chang, "A novel frequent pattern mining algorithm for very large databases in cloud computing environments," in Proc. IEEE Int. Conf. Granular Comput. (GrC), Kaohsiung, Taiwan, 2011, pp. 399–403.
[4] 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.

[5] 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, 2012, pp. 85–94.

[6] S. Hong, Z. Huaxuan, C. Shiping, and H. Chunyan, "The study of improved FP-growth algorithm in MapReduce," in Proc. 1st Int. Workshop Cloud Comput. Inf. Security, Shanghai, China, 2013.

[7] J. Choi, C. Choi, K. Yim, J. Kim, and P. Kim, "Intelligent reconfigurable method of cloud computing resources for multimedia data delivery," Informatica, vol. 24, no. 3, 2013.

[8] 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.

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

[10] J. Zhang, X. Zhao, S. Zhang, S. Yin, and X. Qin, "Interrelation analysis of celestial spectra data using constrained frequent pattern trees," Knowl.-Based Syst., vol. 41, Mar. 2013.

[11] Yaling Xun, Jifu Zhang, and Xiao Qin, "FiDoop: Parallel Mining of Frequent Itemsets Using MapReduce", IEEE Transactions on Systems, Man and Cybernetics: Systems, Volume 46, Issue 3, March 2016.