Impact Factor (SJIF): 4.542

International Journal of Advance Research in Engineering, Science & Technology

e-ISSN: 2393-9877, p-ISSN: 2394-2444

Volume 4, Issue 5, May-2017

SURVEY ON FREQUENT ITEMSET MINING ON HADOOP CLUSTER USING FIDOOP-DP TECHNIQUE

[1]SUSHMA S G, [2]Dr. MOHAMMED RAFI
[1]PG student, Dept of CSE, UBDTCE, Davengere
[2]Professor Dept of CSE, UBDTCE, Davengere

Abstract- The aim of conventional parallel mining algorithms for mining frequent itemsets is to balance load among computing nodes by equally partitioning data. Here for a given large dataset the strategy of data partitioning in existing system suffer mining overhead and high communication that is induced by redundant transaction transmitted among computing nodes. This problem is addressed by developing a data partitioning method called Fidoop-dp using mapreduce programming model he goal of Fidoop is to increase the performance of parallel frequent itemset mining on hadoop cluster. Fidoop-dp uses a hashing technique like placing highly the most similar transaction in to a data partition to improve the data locality without creating an excessive number of redundant transaction.

I INTRODUCTION

The conventional parallel frequent itemset mining techniques are focused on load balancing and here the data is equally portioned and also distributed among computing nodes of a cluster. Here the lack of analysis of correlation among the data leads to poor data locality. The absence of data collocation will make increase in the data shuffling cost and it also reduces the effectiveness of data partitioning. So therefore the data partitioning must pay attention towards network and computing loads and also the issues of load balancing. So a parallel Frequent miming approach called Fidoop-dp using the map reduce programming model. The basic idea of Fidoop-dp is to group a highly relevant transaction in to a data partition so that the number of transaction is significantly slashed.

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. 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,, FP-Growth-based as well as other classic association rule mining . Hadoop is an open source implemen-tation of the MapReduce programming model . 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 FIM built in Hadoop aim at balancing computation load by equally distributing data among nodes. How-ever, 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, in which a large data-set is partitioned across a Hadoop cluster's data nodes in a way to improve data locality.

II PRELIMINARIES

In this section, we first briefly review Frequent itemset mining. Then, to facilitate the presentation of FiDoop-DP, we introduce the Map-Reduce programming framework.

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 item-set X has supports if s% of transactions contain

the itemset. The purpose of FIM is to identify all frequent itemsets whose support is greater than the minimum sup-port. 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 effi-cient and fault tolerant workload distribution in large clus-ters. 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 inter-mediate 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 . At the heart of HDFS is a sin-gle 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.

III DATA PARTITIONG METHODS

FIM is a multi-stage parallel process, where redundant transactions transmission and redundant mining tasks occur in the second MapReduce job. It is a grand challenge to avoid these downsides by using traditional grouping strategies and default partition-ing function. And transferring redundant transactions is a main reason behind high network load and redundant min-ing cost. To solve this problem, we propose to partition transactions by considering correlations among transactions and items prior to the parallel mining process.

1. Distance Metric

In FIM, a good partitioning strategy should cluster similar data objects to the same partition. Sim-ilarity 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 two data 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 dis-tance among transactions is measured using the Jaccard similarity among these sets.

2. K- means Selection of Pivots

Intuitively, selecting pivots directly affects t he uniformity coefficient of the remaining objects f or voronoi diagram based partitioning. I n particular, w e employ t he K-means-based selection strategy (see [19]) to 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 [20], [21]. 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) , in which 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 k-means algorithm. It is worth mentioning that the selection of initial pivots (a.k.a., seeds) plays a critical role in clustering performance. Thus, k-means++ [22]- an extension of k-means, is adopted to conduct pivots selection. After the k data clusters are generated, we choose the center point of each cluster as a pivot for the Voronoi diagram-based data partitioning.

3. Partitioning Strategies

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 founda-tion for LSH.

IV RELATED WORK

Data Partitioning in Map Reduce Partitioning in databases has been widely studied, for both single system servers and distributed storage systems . The existing approaches typically produce possible ranges or hash parti-tions, which are then evaluated using heuristics and cost models. These schemes offer limited support for OLTP workloads or query analysis in the context of the popular MapReduce programming model. In this study, we focus on the data partitioning issue in MapReduce.

High scalability is one of the most important design goals for MapReduce applications. Unfortunately, the partition-ing techniques in existing MapReduce platforms (e.g., Hadoop) are in their infancy, leading to serious perfor-mance problems.

Recently, a handful of data partitioning schemes have been proposed in the MapReduce platforms. Xie et al. developed a data placement management mechanism for heterogeneous Hadoop clusters. Their mechanism partitions data fragments to nodes in accordance to the nodes' processing speed measured by computing ratios. In addition, Xie et al . designed a data redistribution algorithm in HDFS to address the data-skew issue imposed by dynamic data insertions and deletions. Hadoop is a Hadoop's lightweight extension, which h i s designed to identify related data files followed by a modified data placement policy to co-locate c copies of t hose related fi les in t he same server.

Hadoop considers t he relevance among files; that is, Co Hadoop is an optimization of HaDoop for multiple file s. A key assumption of the Map Reduce programming model is t hat mappers are completely independent of one another. Vernica et al. broke such an assumption by introducing a n asynchronous communication channel among mappers. This channel enables the mappers to see global states managed in metadata . Such situation-aware mappers (SAMs) can en able MapReduce to flexibly partition the inputs. Apart f rom this, adaptive sampling and partitioning were proposed to p produce balanced partitions for the reducers by sampling mapper out puts and making use of obtained statistics.

Graph and hypergraph partitioning have been used to guide data partitioning in parallel computing. Graph-based partitioning schemes capture data relationships. For exam-ple, Ke et al. applied a graphic-execution-plan graph (EPG) to perform cost estimation and optimization by analyzing various properties of both data and computation . Their estimation module coupled with the cost model estimate the runtime cost of each vertex in an EPG, which represents the overall runtime cost; a data partitioning plan is deter-mined by a cost optimization module. Liroz-Gistau et al. proposed the MR Part technique, which partitions all input tuples producing the same intermediate key co-located in the same chunk. Such a partitioning approach minimizes data transmission among mappers and reducers in the shuffle phase . The approach captures the relationships between input tuples and intermediate keys by monitoring the execution of representative workload. Then, based on these relationships, their approach applies a min-cut k-way graph partitioning algorithm, thereby partitioning and assigning the tuples to appropriate fragments by modeling the workload with a hyper graph. In doing so, subsequent MapReduce jobs take full advantage of data locality in the reduce phase. Their partitioning strategy suffers from adverse initialization overhead.

Application-Aware Data Partitioning Various efficient data partitioning strategies have been pro-posed to improve the performance of parallel computing systems. For example, Kirsten et al . developed two general partitioning strategies for generating entity match tasks to avoid memory bottlenecks and load imbalances [37]. Taking into account the characteristics of input data, Aridhi et al. proposed a novel density-based data partitioning technique for approximate large-scale frequent subgraph mining to balance computational load among a collection of machines. Kotoulas et al. built a data distribution mechanism based on clustering in elastic regions [38]. Traditional term-based partitioning has limited scalability due to the existence of very skewed frequency distributions among terms.

Load-balanced distributed clustering across networks and local clustering are introduced to improve the chance that triples with a same key are collocated. These self-organizing approaches need no data analysis or upfront parameter adjustments in a priori. Lu et al. studiedk nearest neighbor join using MapReduce, in which a data partitioning approach was designed to reduce both shuffling and computational costs [19]. In Lu's study, objects are divided into partitions using a Voronoi diagram with carefully selected pivots. Then, data partitions (i.e., Voronoi cells) are clustered into groups only if distances between them are restricted by a specific bound. In this way, their approach can answer the k-nearest-neighbour join queries by simply checking object pairs within each group.

FIM for data-intensive applications over computing clus-ters has received a growing attention; efficient data parti-tioning strategies have been proposed to improve the performance of parallel FIM algorithms. A MapReduce-based Apriori algorithm is designed to incorporate a new dynamic partitioning and distributing data method to improve mining performance [39]. This method divides input data into relatively small splits to provide flexibility for improved load-

balance performance. Moreover, the master node doesn't distribute all the data once; rather, the rest data are distributed based on dynamically changing workload and computing capability weight of each node.

Similarly, Jumbo [40] adopted a dynamic partition assign-ment technology, enabling each task to process more than one partition. Thus, these partitions can be dynamically reassigned to different tasks to improve the load balancing performance of Pfp [11]. Uthayopas et al. investigated I/O and execution scheduling strategies to balance data processing load, thereby enhancing the utilization of a multi-core cluster system supporting association-rule mining. In order to pick a winning strategy in terms of data-blocks assign-ment, Uthayopas et al. incorporated three basic placement policies, namely, the round robin, range, and random place-ment. Their approach ignores data characteristics during the course of mining association rules.

IV CONCLUSION

To mitigate high communication and reduce computing cost in MapReduce-based FIM algorithms, we developed FiDoop-DP, which exploits correlation among transactions to partition a large dataset across data nodes in a Hadoop cluster. FiDoop-DP is able to (1) partition transactions with high similarity together and (2) group highly correlated fre-quent items into a list. One of the salient features of FiDoop-DP lies in its capability of lowering network traffic and com-puting load through reducing the number of redundant transactions, which are transmitted among Hadoop nodes. FiDoop-DP applies the Voronoi diagram-based data parti-tioning technique to accomplish data partition, in which LSH is incorporated to offer an analysis of correlation among transactions. At the heart of FiDoop-DP is the second MapReduce job, which (1) partitions a large database to form a complete dataset for item groups and (2) conducts FP-Growth processing in parallel on local partitions to gen-erate all frequent patterns. Our experimental results reveal that FiDoop-DP significantly improves the FIM perfor-mance of the existing Pfp solution by up to 31 percent with an average of 18 percent.

REFERENCES

[1] *M. J. Zaki*, "Parallel and distributed association mining:A survey," IEEE Concurrency , vol. 7, no. 4, pp. 14–25, Oct. 1999.

[2] *I. Pramudiono and M. Kitsuregawa,* "Fp-tax: Tree structure based generalized association rule mining," in Proc. 9th ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery, 2004, pp. 60–63.

[3] *J. Dean and S. Ghemawat,* "Mapreduce: Simplified data processing on large clusters," AC M C om mu n, vol. 51, no. 1, pp. 107–113, 2008.

[4] T. Kirsten, L. Kolb, M. Hartung, A. Groß, H. K €opcke, and E. Rahm, "Data partitioning for parallel entity matching," Proc. VLDB Endowment , vol. 3, no. 2, pp. 1–8, 2010.

[5] S. Kotoulas, E. Oren, and F. Van Harmelen, "Mind the data skew: Distributed inferencing by speeddating in elastic regions," in Proc. 19th Int. Conf. World Wide Web, 2010, pp. 531–540.

[6] L. Li and M. Zhang, "The strategy of mining association rule based on cloud computing," in Proc. Int. Conf. Bus. Comput. Global Inform., 2011, pp. 475–478.

[7] S. Groot, K. Goda, and M. Kitsuregawa, "Towards improved load balancing for data intensive distributed computing," in Proc. ACM Symp. Appl. Comput., 2011, pp. 139–146.

[8] M. Z. Ashrafi, D. Taniar, and K. Smith, "ODAM: An optimized distributed association rule mining algorithm," IEEE Distrib. Syst. Online , vol. 5, no. 3, p. 1, Mar. 2004.

[9] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in Proc. 2nd USENIX Conf. Hot Topics Cloud Comput., 2010, p. 10.

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

[11] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 7, pp. 881–892, Jul. 2002.