International Innovative Research Journal of Engineering and Technology

ISSN NO: 2456-1983

International Conference on Emerging Innovation in Engineering and Technology
ICEIET-2017

Fidoop-Dp: Data Partitioning In Frequent Itemset Mining On Hadoop Clust

[1]DR.P.SHUNMUGAPRIYA [2]R.MARAGATHAM
[1]Dean of school of computing, Christ Institute of Technology, Puducherry.
[2]PG Scholar, Department of CSE, Christ Institute of Technology, Puducherry.
pshunmugapriya@gmail.com roja.maragatham@gmail.com

ABSTRACT
Traditional parallel algorithms for mining frequent itemsets aim to balance load by equally partitioning data among a group of computing nodes. We start this study by discovering a serious performance problem of the existing parallel Frequent Itemset Mining algorithms. Given a large dataset, data partitioning strategies in the existing solutions suffer high communication and mining overhead induced by redundant transactions transmitted among computing nodes. We address this problem by developing a data partitioning approach called FiDoop-DP using the MapReduce programming model. The overarching goal of FiDoop-DP is to boost the performance of parallel Frequent Itemset Mining on Hadoop clusters. At the heart of FiDoop-DP is the Voronoi diagram-based data partitioning technique, which exploits correlations among transactions. Incorporating the similarity metric and the Locality-Sensitive Hashing technique, FiDoop-DP places highly similar transactions into a data partition to improve locality without creating an excessive number of redundant transactions. We implement FiDoop-DP on a 24-node Hadoop cluster, driven by a wide range of datasets created by IBM Quest Market-Basket Synthetic Data Generator. Experimental results reveal that FiDoop-DP is conducive to reducing network and computing loads by the virtue of eliminating redundant transactions on Hadoop nodes. FiDoop-DP significantly improves the performance of the existing parallel frequent-pattern scheme by up to 31% with an average of 18%.

Keywords : Frequent Itemset Mining, Parallel Data Mining, Data Partitioning, MapReduce Programming Model, Hadoop Cluster

I. INTRODUCTION

Traditional parallel Frequent Itemset Mining techniques (a.k.a., 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, 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. 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.

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 (e.g., Aprioribased, FP-Growth-based , as well as other classic association rule mining). Hadoop is an open source implementation of the MapReduce programming model In this paper[3], 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. 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 FiDoop-DP, a parallel FIM technique, in which a large dataset is partitioned across a Hadoop cluster's data nodes in a way to improve data locality.

II. LITERATURE SURVEY

Fidoop:parallel mining of frequent items using mpreduce

ISSN NO: 2456-1983

Existing parallel mining algorithms for frequent itemsets is not efficient.To solve the 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.

Frequent Set Mining For Streaming Mixed And Large Data [2]

Frequent set mining is a well researched problem due to its application in many areas of data mining such as clustering, classification and association rule mining. Most of the existing work focuses on categorical and batch data and do not scale well for large datasets. In this work, introduce a discretization methodology to find meaningful bin boundaries when item sets contain at least one continuous attribute. An update strategy to keep the frequent items relevant in the event of concept drift, and a parallel algorithm to find these frequent items. Our approach identifies local bins per itemset, as a global discretization may not identify the most meaningful bins.

Efficient Apriori Based Algorithms For Privacy Preserving Frequent Itemset Mining [3]

Frequent Itemset Mining as one of the principal routine of data analysis and a basic tool of large scale information aggregation also bears a serous interest in Privacy Preserving Data Mining. In this paper Apriori based distributed, privacy preserving Frequent Itemset Mining algorithms are considered.

Our secure algorithms are designed to fit in the Secure Multiparty Computation model of privacy preserving computation.

III. EXISTING SYSTEM

Existing parallel Frequent Itemset Mining algorithms given a large dataset, data partitioning strategies in this the solutions suffer high communication. And mining overhead induced by redundant transactions transmitted among computing nodes. In this paper [2],the partitioning techniques in this MapReduce platforms are in their infancy, leading to serious performance problems.

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. Existing data partitioning solutions of FIM 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.

Disadvantages

- Parallel algorithms lack a mechanism that enables
- Automatic parallelization,
- Load balancing,
- Data distribution, and
- Fault tolerance on large computing clusters.

IV. PROPOSED SYSTEM

FiDoop-DP using the MapReduce programming model is proposed. The goal of FiDoop-DP is to boost the performance of parallel Frequent Itemset Mining on Hadoop clusters. It is the Voronoi diagram-based data partitioning technique, which exploits correlations among transactions. It places highly similar transactions into a data partition to improve locality without creating an excessive number of redundant transactions. the proposed FiDoop- DP, We generate synthetic datasets using the IBM Quest Market-Basket Synthetic Data Generator , which can be flexibly configured to create a wide range of data sets to meet the needs of various test requirements.

Application-Aware Data Partitioning

Various efficient data partitioning strategies have been proposed 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 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

Data Characteristic Dimensionality:

FiDoop-DP to efficiently reduce the number of redundant transactions. In contrast, a dataset with high dimensionality has a long average transaction length; therefore, data partitions produced by FiDoop-DP have no distinct discrepancy. Redundant transactions are likely to be formed for partitions that lack distinct characteristics. Consequently, the benefit offered by FiDoop-DP for high dimensional datasets becomes insignificant.

Data Correlation:

FiDoop-DP judiciously groups items with high correlation into one group and clustering similar transactions together. In this way, the number of redundant transactions kept on multiple nodes is substantially reduced. Consequently, FiDoop-DP is conducive to cutting back both data transmission traffic and computing load.

International Innovative Research Journal of Engineering and Technology

ISSN NO: 2456-1983

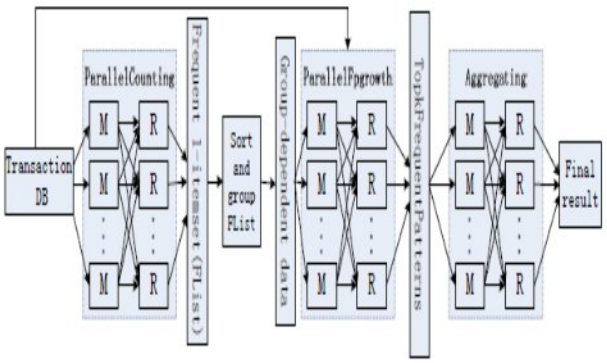

Figure 1 : System Architecture

In Figure 1 : 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 FList: 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 FList. 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. First, the Mappers divide all the items in FList into Q groups. The list of groups is referred to as group list or GList, where each group is assigned a unique group ID (i.e., Gid). Then, the transactions are partitioned into multiple groups according to GLists. That is, each mapper outputs one or more key-value pairs, where a keys is a group ID and its corresponding value is a generated group-dependent transaction. • Reducer - FP-Growth on group-dependent partitions. lo- cal 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.

Advantages

- •Automatic parallelization,
- •Load balancing,
- •Data distribution, and
- •Fault tolerance on large computing clusters

Nearest Neighbor Classifier: K-Nearest Neighbor Classifier (Knn) And Its Modifications

It is a majority of class theorem for the newly came unclassified document where k denotes the number of already classified documents and k is not the multiple of number of classes.

(i) Standard KNN- k is fixed. Weight factor is not considered.

(ii) Time consuming. k-variable KNN- Improved k-variable KNN, Basic kvariable KNN, Weighting KNN are good if they are combined into one 'Flexible KNN' algorithm which switches the algorithms according to k value available but again it is somewhat complex also not feasible real time sentiment analysis

V. CONCLUSION AND FUTURE ENHANCEMENTS

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 frequent items into a list. One of the salient features of FiDoop-DP lies in its capability of lowering network traffic and computing load through reducing the number of redundant transactions, which are transmitted among Hadoop nodes. FiDoop-DP applies the Voronoi diagrambased data partitioning 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 generate all frequent patterns. Our experimental results reveal that FiDoop-DP significantly improves the FIM performance of the existing Pfp solution by up to 31% with an average of 18%. We introduced in this study a similarity metric to facilitate data-aware partitioning. As a future research direction, we will apply this metric to investigate advanced loadbalancing strategies on a heterogeneous Hadoop cluster.

References

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

[2] I. Pramudiono and M. Kitsuregawa, “Fp-tax: Tree structure based generalized association rule mining,” in Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery. ACM, 2004, pp. 60–63.

[3] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[4] S. Sakr, A. Liu, and A. G. Fayoumi, “The family of mapreduce and large-scale data processing systems,” ACM Computing Surveys (CSUR), vol. 46, no. 1, p. 11, 2013.

[5] 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

ISSN NO: 2456-1983

Communication, ser. ICUIMC '12. New York, NY, USA: ACM, 2012, pp. 76:1–76:8.

[6] 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.

[9] L. Zhou, Z. Zhong, J. Chang, J. Li, J. Huang, and S. Feng, "Balanced parallel fp-growth with mapreduce," in
[7] M. Riondato, J. A. DeBrabant, R. Fonseca, and E. Upfal, "Parma: a parallel randomized algorithm for approximate association rules mining in mapreduce," in Proceedings of the 21st ACM international conference on Information and knowledge management. ACM, 2012, pp. 85–94.
[8] C. Lam, Hadoop in action. Manning Publications Co., 2010.

Information Computing and Telecommunications (YC-ICT), 2010 IEEE Youth Conference on. IEEE, 2010, pp. 243–246.

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

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