EURASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES
Innovative Academy Research Support Center
UIF = 8.3 | SJIF = 5.916 www.in-academy.uz

PERFORMANCE ANALYSIS OF ASSOCIATION RULE MINING ALGORITHMS USING HADOOP

Rakhimova L.S
Assistant of the Department of "Software Engineering", Urgench branch of Tashkent University of Information Technologies named after Muhammad al-Kharezmi, Khorezm, Uzbekistan. " 05.00.00 - Texnika fanlari, Axborot texnologiyalari sohalari", +998975135152
https://doi.org/10.5281/zenodo.7478791

ARTICLE INFO

Received: 13th December 2022
Accepted: 21th December 2022
Online: 23th December 2022

KEY WORDS

Association rule, Apriori, Hadoop, Map Reduce, Big data.

ABSTRACT

Association rule mining has been a very important method in the field of data mining. Apriori algorithm is a classical algorithm for association rule mining. In the big data environment, the traditional Apriori algorithm has been unable to meet the needs of mining. In the paper, the parallelization of the Apriori algorithm is implemented based on the Hadoop platform and the Map Reduce programming model. On the basis, the algorithm is further optimized by using the idea of transaction reduction. Experimental results show that the proposed algorithm can be better to meet the requirements of big data mining and efficiently mining frequent itemsets and association rules from large dataset.

I. INTRODUCTION

We are surrounded with excessive amount of digital data but ravenous for potentially precise information. Data mining is the technique that finds hidden insight and unknown patterns from massive database, which are used as useful knowledge for decision making. Association Rule Mining (ARM) [1] is one of the most important functionality of data mining that comprises of two tasks; finding frequent itemsets and finding interesting correlation between set of frequent items. An itemset is said to be frequent if its support is greater than or equal to a user defined minimum support threshold. Support of an itemset is the percentage of transactions containing that itemset in database. The Apriori algorithm proposed by R. Agrawal and R. Srikant [2] is the most widely used algorithm for mining frequent itemset. Various data structures and a number of sequential and parallel algorithms hav e been designed to enhance the performance of Apriori algorithm.

Big Data [3] technologies create a biggest hype just after its emergence. A new parallel and distributed computing paradigm has been introduced which is largely scalable and does not require high-end machines. Hadoop is such a large-

scale distributed batch processing infrastructure for parallel processing of big data on large cluster of commodity computers [4]. Map Reduce is an efficient and scalable parallel programming model of Hadoop that process large volumes of data in parallel and distributed fashion. Traditional tools and techniques of data mining are not scalable and efficient to manage big data. Recent advances are porting data mining algorithms on this new paradigm. Many authors have re-designed and implemented the Apriori algorithm on Map Reduce

framework in an efficient way but the impact of data structures on the efficiency of Map Reduce based Apriori algorithm have not been yet evaluated. Data structures are the integral in designing of any algorithm. A well-organized data structure significantly reduces the time and space complexity. Apriori algorithm finds the frequent itemsets by generating a large number of candidate itemsets. Candidates are the itemsets containing all potentially frequent itemsets.

II. BACKGROUND

Association Rule Mining

Association Rules reveals the patterns or associations that have applications in many domains like marketing, decision making and inventory management. It is also used in areas like Market Basket Analysis, Web Mining, studying patterns of Biological databases, analyze Population and Economic Census, Medical diagnosis, and Scientific Data Analysis. Association rules shows itemsets that occur frequently in transactions and the generated rules are probabilistic in nature.

The mining or formation of association rules consists of two steps:

- Frequent Itemset Mining: finding the itemsets that occurs frequently in the database based on support.
- Association Rule Generation: generating strong rules from the frequent itemsets based on confidence.

Usually, an Association Rule Mining Algorithm results in the formation of a large number of association rules and it is difficult for users to validate all the rules manually. Therefore, only interesting rules or non-repeating rules are to be generated.

Apriori

Apriori is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis. There are three types of Parallel versions of Apriori algorithms:

- Count Distribution: Local processing and global verification
- Data Distribution: Mutual Exclusive Candidate items are distributed.
- Candidate Distribution: Independent Candidate items set for each processor.

MapReduce Model

MapReduce is one of the earliest and best known models in parallel and distributed (cluster) computing area, created by Google in 2004, based on C++ language. It is a programming model and associated implementation for processing and generating large data sets in a massively parallel and distributed manner [5]. One of the attractive qualities of MapReduce model is its simplicity: a MapReduce application (job) consists only of two functions, Map and Reduce functions. Developers write the map function that processes a key/value pair to generate a set of intermediate key/value pairs, and the reduce function that merges all intermediate values associated with the same intermediate key. Many data mining areas such as:

association rule, clustering and classification algorithms have been implemented based on MapReduce model [6, 7, 8].

Hadoop

Hadoop is an open source framework, created as an open source implementation of Google MapReduce architecture, based on Java language, sponsored by Apache Software Foundation. "Hadoop MapReduce is an open source software framework for writing applications which process vast amounts of data (multi-terabyte datasets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner [4]." The base Apache Hadoop framework is composed of the following modules:

Hadoop Common – contains libraries and utilities needed by other Hadoop modules;

Hadoop Distributed File System (HDFS) – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;

Hadoop YARN – a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users' applications;

Hadoop MapReduce – an implementation of the MapReduce programming model for large scale data processing.

III. AN IMPLEMENTATION AND IMPROVEMENT OF APRIORI ALGORITHM BASED ON HADOOP

Hadoop implementation of Apriori algorithm

There are two general ideas for Hadoop implementation in Apriori. In this paper, we implemented it by iterative MapReduce, because the default data block size for HDFS is 64 MB. If implemented in a different way, it means that each node needs to process 64 MB of data separately, and a large number of local frequent itemsets can be generated. Although Hadoop allows to customize the size of file block, this change can have an impact on Hadoop's parallelism and scalability. It's not a good choice.

The following steps are described for the program implementation.

Step1 : The mining of frequent itemsets requires first to determine the actual support count and this is determined by the total number of transactions and the percentage of input support. In experiments, however, the file is used instead of the database to store the transaction set. Therefore, a single Job is required to complete the statistics of the total number of transactions. After the statistics are complete, the minimum support count is calculated and placed in the global configuration object.

Step2: The process of mining frequent itemsets1 is similar to the word count process. The difference is that the Reducer is required to determine whether the count of each entry satisfies the minimum support and outputs the entries that satisfy the minimum support. The process also requires a new Job for the main process.

Step3: This step is the core of the whole algorithm. First of all, we need to read the frequent itemsets1 obtained in the previous step then generates the candidate itemsets2 after the connection. Since the candidate itemsets2 cannot be pruned, only the join step is required. Then, count and select the support of the candidate itemsets2. Mapper gets the path of the candidate itemsets2 from the global configuration object and reads the entries in the file into memory. Subsequently, each transaction is read in rows in the map function, and the candidate itemsets2 contained in the transaction are output to the Combiner. Combiner first

summarizes the entries of the same local key value and outputs the local summary results to Reducer.

Reducer makes a final summary of all the results and determines whether the support count of the each item satisfies the minimum support count. If satisfied, the entries and their support are output to HDFS (Hadoop Distributed File System). After the Job process is completed, the main process reads the k-1 frequent itemsets in the output path of the last frequent itemsets and determines whether the k-1 frequent itemsets are empty. And, if so, ends the mining. If not, k-1 frequent itemsets make self connection and prune by a priori nature. Then, a Job is used to count the support of the frequent itemsets and the process is iterated until the frequent itemsets are empty.

An improved algorithm based on transaction reduction

The last section describes the basic idea of the implementation of Apriori algorithm in Hadoop platform. In this way, the scanning process of database is parallelized and database scanning is one of the main bottlenecks of Apriori algorithm. On the basis of the basic implementation, we can also improve the implementation of the algorithm. This paper uses the transaction reduction method to reduce the number of scanning database transactions. The idea of transaction reduction is also based on a property of frequent itemsets: transactions that do not contain any k-1 frequent itemsets can not contain k frequent itemsets. Therefore, these transactions can be marked during database scanning. And the number of transactions that need to be scanned is reduced. So it improves the efficiency of mining.

The first problem that needs to be solved is how to uniquely identify each transaction record based on transaction reduction algorithms. In HDFS, each file is stored as a block of 64 MB and each block has a unique URL. During MapReduce execution, each Mapper needs to process a split separately. When it reads a transaction record by line, the key value is the number of offset bytes that the row records in the file. For this record, this key value can be its only identifier in the split. So, the split of the URL and the key value of the transaction record can be uniquely identified. According to this strategy, the improvement focuses on the Mapper implementation logic. Mapper first needs to get the split of the URL and deposit it into a member variable in Mapper. At the same time, according to split's URL and the path, find the file that stores its rejection list, then reads the rejection list into a HashSet. When the map function counts the candidate set, if it is found that the transaction does not contain any candidate itemsets, then it is added to the latest rejection list. Finally, the new rejection list is appended to the rejection file in the Mapper function of cleanup for use in the next scan.

IV. CONCLUSION

A parallelization of Apriori algorithm is proposed based on Hadoop platform in this paper. The parallel scanning through the transaction itemsets can greatly improve the efficiency of the algorithm. At the same time in order to reduce the database scanning consumption, using the transaction optimization algorithm to achieve further reduction of thought improve the efficiency of the algorithm. The experiments show that the parallel Apriori algorithm in this paper has less time consumption than traditional apriori algorithm and can mine frequent itemsets more quickly.

References:

1. Agrawal, R., Imielinski, T. and Swami, A. 1993. Mining Association Rules between Sets of Items in Large Databases. In ACM SIGMOD Conf. Management of Data, Washington, D.C., 207–216.
2. Agrawal, R. and Srikant, R. 1994. Fast Algorithms for Mining Association Rules. In Proceedings of the Twentieth International Conference on Very Large Databases, Santiago, Chile, 487–499.
3. Ward, J. S. and Barker, A. Undefined By Data: A Survey of Big Data Definitions. http://arxiv.org/abs/1309.5821v1. Retrieved Sept. 2015.
4. Apache Hadoop. http://hadoop.apache.org
5. Dean J. & Ghemawat S. (2004). MapReduce: Simplified Data Processing on Large Clusters. Proc. of the 6 th Symposium on Operation Systems Designing and Implementation (OSDI '04). San Francisco, CA, Google Inc.: 1- 13.
6. Li H., Wang Y., Zhang D., Zhang M. & Chang E.Y. (2008). PFP: Parallel FP-Growth for Query Recommendation. Proceeding of the 2008 ACM conference on Recommender systems (RecSys '08). New York, NY, USA, ACM: 107 – 114.
7. He Q., Zhuang F., Li J. & Shi Z. (2010). Parallel Implementation of classification algorithms based on MapReduce. Proc. of the 5 th International Conference on Rough Set and Knowledge Technology (RSKT '10). Berlin, Heidelberg, Springer: 655 – 662.