Research on Apriori Algorithm Optimization of Cloud Computing and Big Data in Software Engineering

Wang Rui

Yunnan Engineering Vocational College, Anning, Yunnan, 650304

Keywords: Cloud computing; data mining; Apriori algorithm; MapReduce; frequent item sets

Abstract: With the continuous expansion of data information, it is more and more difficult to extract effective data information from data analysis. Traditional data analysis algorithms can no longer meet the needs of big data analysis. The rise of Cloud computing provides a new solution to this problem. This paper studies the Cloud computing technology and data mining and analyses the Apriori algorithm. Based on its limitations, it proposes an optimization scheme and introduces the MapReduce model in Cloud computing to achieve parallelization. A MapReduce-based frequent item set mining method is proposed to improve the efficiency of the algorithm and reduce the overhead required for algorithm execution.

1. Introduction

In the face of the ever-increasing number of Internet data, it is difficult to accurately obtain the data required by organizations and individuals on the Internet. Therefore, data mining and designing of Internet data has become a hot topic [1]. This paper first briefly introduces cloud computing and data mining technologies, then introduces and analyzes the Apriori algorithm, and proposes an improved method of the algorithm, namely: Apriori algorithm based on MapReduce, introducing the cloud computing functions Map and Reduce into the Apriori algorithm, respectively. Algorithm, designing frequent item set mining methods based on Map/Reduce to improve cloud computing data mining capabilities [2].

Therefore, it is necessary to combine data mining technology with Internet cloud computer data technology, and adopt appropriate data mining algorithms on cloud computing platforms to achieve rapid cloud mining of Internet computing data [3]. This paper uses the Hadoop cloud computing framework as a data mining platform. This platform is characterized by high reliability, strong fault tolerance, and strong scalability. It is an open source cloud computing platform.

2. Theoretical part

2.1 Apriori algorithm

Association rules are rules that reflect the relationships and dependencies between things. Apriori algorithm is one of the more classical algorithms in many association algorithms. The Apriori algorithm is an association rule algorithm and a breadth-first algorithm [4]. It uses a hierarchical search strategy from the bottom up to calculate the number of occurrences of the items in the database, and then cuts the items according to the user-defined minimum support to obtain the minimum support items. Generate new candidate sets and filter them.

The main property of the Apriori algorithm is that the assumed item set {A, C} is a frequent itemset, then {A}, {C} is also a frequent itemset; assuming the itemset {D} is not a frequent item set, then {A, D} And {C, D} is not a frequent item set. That is, the subset of frequent itemsets must be frequent, and the superset of infrequent itemsets must be infrequent. These properties of Apriori algorithm can improve the efficiency of generating frequent itemsets to some extent.

The Apriori algorithm can be divided into two steps for discovering association rules: calculating candidate sets, obtaining frequent itemsets, and constructing strong association rules that satisfy the minimum required confidence min_conf using frequent itemsets[5]. First of all, during the frequent

item discovery process, iterative methods are needed to continuously repeat the scan data set. Each candidate item is counted, compared, and frequently itemsets are generated. Then pruning is performed to generate the candidate item set process until no more can be found. Large frequent itemsets. Secondly, an association rule is generated to generate all nonempty subsets for each frequent item set L. For each nonempty subset S of L, if |L/S|>min_conf, the rule L→LS is output, and LS is represented in the item set. L removes the S-item set of items.

2.2 MapReduce programming model

MapReduce is a distributed programming model based on cloud computing and big data. It can perform collection-parallel distribution of data values. The principle of its programming model is to use an input key-value pair set to generate an output key-value pair set. MapReduce programming is divided into two phases, Map mapping and Reduce reduction.

The function map splits the input data in the MapReduce model framework, decomposes the large data set into several small data sets, and allocates the decomposed small data sets to the tasks of the Map function [6]. The Map function is allocated to the small data sets. The keywords and data types (ie Key/value pairs) are divided and calculated in parallel, and then the resulting data sets with the same keywords and data types are classified and passed to the Reduce reduction function for classification.

The function Reduce reduction reorganizes and merges the small data sets analyzed by the Map mapping function. The data sets with the same keyword and data type are merged together to form a new set.

3. MapReduce-based prior algorithm design

3.1 Frequent itemset generation based on MapReduce model

MapReduce model for frequent itemsets mining process shown in Figure 1.

The basic steps are as follows:

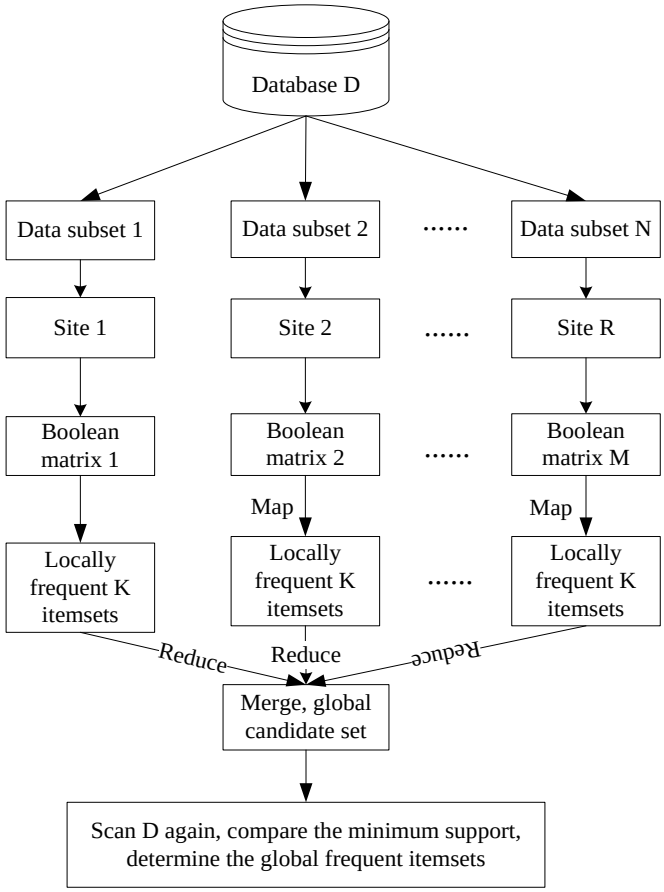

Figure 1. Flowchart for mining frequent itemsets based on MapReduce model

1) First, the transaction database D is divided into a number of data subsets of equal size and disjointness, and each data subset is sent to R sites in parallel (R≤N); then matrix conversion is performed on R sites., each generates a Boolean matrix.

2) Finally scan D again, compare with minimum support count min_sup, and finally determine the global frequent K itemset L_k. It can be seen that, through the idea of dataset segmentation, each small block calculates the supportability of the candidate set separately, and realizes group statistics. Therefore, the generation of local frequent K itemsets in each small block is relatively opposite and thus increases to some extent. The efficiency of the algorithm reduces the communication between nodes. The number of scans for transaction database D is only two times, which greatly reduces the overhead required for algorithm execution.

3.2 Algorithm implementation process

The following is a case study to illustrate the feasibility of the improved algorithm. Suppose a transaction database D has a total of 10 records, as shown in Table 1. The size of the data block can be set as M=3, and the user-defined minimum support count is 2.

The process of generating frequent item sets is as follows:

Table 1. Data in a transaction database

Transaction ID	Item set
T_1	ABD
T_2	BD
T_3	ABC
T_4	AD
T_5	ABE
T_6	ABCE
T_7	ADE
T_8	ABCD

1) M=3, that is, D is divided into three data blocks, namely $D_1=\{T_1,T_2,T_3\}$, $D_2=\{T_4,T_5,T_6\}$, $D_3=\{T_7,T_8,T_9,T_{10}\}$.

2) The data blocks D_1, D_2, and D_3 are sent to three work sites R_1, R_2, and R_3, respectively.

3) Convert to Boolean matrices, which are M_1, M_2, and M_3, as follows:

$$M_1=\begin{array}{c}\begin{matrix}T_1 & T_2 & T_3\end{matrix}\\ \begin{pmatrix}1&0&1\\1&1&1\\0&1&0\\1&0&0\\1&1&0\end{pmatrix}\end{array}\begin{matrix}A\\B\\C\\D\\E\end{matrix},\quad M_2=\begin{array}{c}\begin{matrix}T_4 & T_5 & T_6\end{matrix}\\ \begin{pmatrix}1&0&1\\0&1&0\\0&1&0\\1&1&0\\0&1&0\end{pmatrix}\end{array}\begin{matrix}A\\B\\C\\D\\E\end{matrix},\quad M_3=\begin{array}{c}\begin{matrix}T_7 & T_8 & T_9\end{matrix}\\ \begin{pmatrix}0&0&1\\0&1&1\\1&1&0\\1&0&0\\0&1&1\end{pmatrix}\end{array}\begin{matrix}A\\B\\C\\D\\E\end{matrix} \tag{1}$$

First, determine the inner product of the row vectors in their respective matrices. For example, in the M_1 matrix, it can be seen that the added value of the respective row vectors is greater than 0 and can be preserved; and in the M_2 matrix, the vector values of the 4th and 5th rows are added to 0. Drop directly.

Parse into the <key, value> form, using the Map function to generate a local candidate 1-item set, details as follows:

R1: <{A},2>, <{B},3>, <{C},1>, <{D},1>, <{E},2>

R2: <{A},2>, <{B},2>, <{C},3>

R3: <{A}, 2>, <{B}, 2>, <{C}, 2>, <{D}, 1>, <{E}, 3>

4) By comparing the local minimum support thresholds, the obtained local frequent 1-item sets are as follows:

R1: L11={{A}, {B}, {C}, {D}, {E}}

R2: L12={{A}, {B}, {C}}

R3: L13={{A}, {B}, {C}, {D}, {E}}

4. Algorithm and performance analysis

The traditional Apriori algorithm needs to scan the transaction database D every time it generates a C_k. The larger the size of D, the longer the running time. To improve the algorithm, mining frequent itemsets requires only scanning the transaction database D twice. The number of records, m is the number of partitions, we can see, in the same connection and pruning case, the greater the m, the smaller the time required; the row vector of the matrix using the "and" operation, can quickly calculate support counting, and can reduce the generation of candidate sets, thereby reducing the number of operations in the pruning process; the second scan only requires comparison screening, time is greatly reduced.

5. Summary

Internet cloud computing data mining can improve the comprehensive utilization efficiency of data. As an algorithm of data mining, the Prior algorithm can calculate frequent data sets and mine data association rules, but its work efficiency is low. The cloud computer system files are stored on each node and are automatically partitioned. This paper designs data mining algorithms based on the April algorithm, transforms the data format, introduces the matrix, and combines the frequent item set mining methods of MapReduce to design the algorithm and optimize Apriori.

References

[1] Guo Z, Chi D, Wu J, et al. A new wind speed forecasting strategy based on the chaotic time series modelling technique and the Apriori algorithm[J]. Energy Conversion and Management, 2014, 84: 140-151.

[2] Liu H, Tian H, Li Y, et al. Comparison of four Adaboost algorithm based artificial neural networks in wind speed predictions[J]. Energy Conversion and Management, 2015, 92: 67-81.

[3] Zhang F, Liu M, Gui F, et al. A distributed frequent itemset mining algorithm using Spark for Big Data analytics[J]. Cluster Computing, 2015, 18(4): 1493-1501.

[4] Niu K, Jiao H, Gao Z, et al. A developed apriori algorithm based on frequent matrix[C]//Proceedings of the 5th international conference on bioinformatics and computational biology. ACM, 2017: 55-58.

[5] Harun N A, Makhtar M, Aziz A A, et al. The Application of Apriori Algorithm in Predicting Flood Areas[J]. International Journal on Advanced Science, Engineering and Information Technology, 2017, 7(3): 763-769.

[6] Zhu S. Research on data mining of education technical ability training for physical education students based on Apriori algorithm[J]. Cluster Computing, 2018: 1-8.