The mining algorithm for maximum frequent itemsets based on big data

Bo He, Hongyuan Zhang
School of Computer Science and Engineering, ChongQing University of Technology, 400054
ChongQing, China
E-mail: heboswnu@sina.com

Abstract: The paper proposed a mining algorithm for maximum frequent itemsets based on big data, namely, MFIBD algorithm. MFIBD algorithm makes computer nodes computed local maximum frequent itemsets independently with DMFIA algorithm and mapreduce, then the center node exchanged data with other computer nodes, finally, maximum frequent itemsets were gained by mapreduce. MFIBD required far less communication traffic by the search strategy of top-down and mapreduce. Theoretical analysis and experimental results suggest that MFIBD algorithm is fast and effective.

Keywords: mining algorithm; maximum frequent itemsets; big data; mapreduce.

1. Introduction

There are some traditional algorithms for mining maximum frequent itemsets[1], such as Max-Miner[2], DMFII[3] and DMFIA[4]. There are also some algorithms for mining global frequent itemsets, include PDM, CD, FDM, etc. However, these mining algorithms do not suit mining maximum frequent itemsets from mass data. Most of them adopt Apriori-like algorithm, so that a lot of candidate itemsets are generated and database is scanned frequently. This causes a large amount of communication traffic among nodes. Aiming at these problems, this paper proposes a mining algorithm for maximum frequent itemsets based on big data, namely, MFIBD algorithm.

Big data is a broad term for data sets so large or complex that traditional data processing applications are inadequate. Challenges include analysis, capture, data curation, search, sharing, storage, transfer, visualization, and information privacy. The term often refers simply to the use of predictive analytics or other certain advanced methods to extract value from data, and seldom to a particular size of data set. Accuracy in big data may lead to more confident decision making. And better decisions can mean greater operational efficiency, cost reductions and reduced risk.

Mapreduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster.

Mapreduce has large scale and highly scalable. The massive distributed date mining based on big data is a very important field.

2. Related definition and theorem

2.1 Description of mining of Maximum frequent itemsets

The global transaction database is $DB$, the total number of tuples is $M$. Suppose $P_1, P_2, \ldots, P_n$ are $n$ computer nodes, there are $M_i$ tuples in $DB_i$, if $DB_i (i=1,2,\ldots,n)$ is a part of $DB$ and stores in $P_i$, then $DB = \bigcup_{i=1}^{n} DB_i$, $M = \sum_{i=1}^{n} M_i$.

Mining of maximum frequent itemsets in distributed database can be described as follows: each node $P_i$ deals with local database $DB_i$, and communicates with other nodes, finally, maximum frequent itemsets of global transaction database are gained.

2.2 Related definition

Definition 1 For itemsets $X$, the number of tuples which contain $X$ in local database $DB_i (i=1,2,\ldots,n)$ is defined as local frequency of $X$, symbolized as $X.si$. 
Definition 2 For itemssets \( X \), the number of tuples which contain \( X \) in global database is global frequency of \( X \), symbolized as \( X.s \).

Definition 3 For itemssets \( X \), if \( X.s \geq \min \_sup * M_i (i = 1,2,\ldots,n) \), then \( X \) are defined as local frequent itemssets of \( DB_i \), symbolized as \( F_i \), \( \min \_sup \) is the minimum support threshold.

Definition 4 For itemssets \( X \), if \( X.s \geq \min \_sup * M \), then \( X \) are defined as global frequent itemssets, symbolized as \( F \). If \( \|X\| > k \), then \( X \) symbolized as \( F_z \).

Definition 5 For global frequent itemssets \( X \), if all superset of \( X \) are not global frequent itemssets, then \( X \) are defined as maximum frequent itemssets, symbolized as \( FM \).

2.3 Related theorem

Theorem 1 If itemssets \( X \) are local frequent itemssets of \( DB_i \), then any nonempty subset of \( X \) are also local frequent itemssets of \( DB_i \).

Theorem 2 If itemssets \( X \) are global frequent itemssets, then \( X \) and all nonempty subset of \( X \) are at least local frequent itemssets of a certain local database.

Theorem 3 If itemssets \( X \) are global frequent itemssets, then any nonempty subset of \( X \) are also global frequent itemssets.

Theorem 4 If itemssets \( X \) are maximum frequent itemssets, then \( X \) must be global frequent itemssets.

Theorem 5 If itemssets \( X \) are maximum frequent itemssets, then \( X \) and all nonempty subset of \( X \) are at least local frequent itemssets of a certain local database.

Theorem 6 If itemssets \( X \) are maximum frequent itemssets, then \( X \) are at least the subset of local maximum frequent itemssets in a certain local database.

3. MFIBD algorithm

3.1 Design thoughts of MFIBD algorithm

MFIBD sets one node \( P_0 \) as the center node, other nodes \( P_i \) send local maximum frequent itemssets \( FM_i \) to the center node \( P_0 \). \( P_0 \) gets local maximum frequent itemssets \( FM' = \bigcup_{i=1}^{n} FM_i \). \( FM' \) are pruned by the searching strategy of top-down. Setting of the center node avoids repetitive calculation which caused by local frequent itemssets existing in many nodes. \( FM' \) are pruned by the searching strategy of top-down, Pruning lessens communication traffic.

Each node adopts DMFIA[4] algorithm and mapreduce to compute local maximum frequent itemsets in MFIBD. Adopting FP-tree structure, DMFIA algorithm greatly reduces database scanning times and runtime compared with Apriori-like algorithm.

The acquirement of global frequent items is the first step of MFIBD. \( P_i \) scans \( DB_i \) once and computes local frequency of local items \( E_i \). \( P_i \) collects global frequency of all items \( E_i \) from each node and gets all global frequent items \( E \). Finally, \( E \) is sorted in the order of descending support count. \( P_0 \) sends \( E \) to other nodes \( P_i \).

Using global frequent items \( E \), MFIBD makes each node \( P_i \) construct \( FP\_tree \). \( P_i \) computes local maximum frequent itemssets \( FM_i \) independently by DMFIA algorithm and \( FP\_tree \). Then the center node \( P_0 \) exchanges data with other nodes and combines by the searching strategy of top-down, finally, maximum frequent itemssets are gained. According to theorem 6, maximum frequent itemssets are at least the subset of local maximum frequent itemssets of one local database, hence the union of each node’s local maximum frequent itemssets \( FM_i \) must be the superset of maximum frequent itemssets \( FM \). Computing local maximum frequent itemssets may be carried out asynchronously, and synchronization is implemented only twice. This lessens synchronization.

3.2 Description of MFIBD algorithm

The pseudocode of MFIBD is described as follows.

Algorithm MFIBD

Input: The local transaction database \( DB_i \) which has \( M_i \) tuples and \( M = \sum_{i=1}^{n} M_i \), \( n \) nodes \( P_i (i = 1,2,\ldots,n) \), the center node \( P_0 \), the minimum support threshold \( \min \_sup \).

Output: The maximum frequent itemssets \( FM \).

Methods: According to the following steps.

Step1. /*each node adopts DMFIA[4] algorithm and mapreduce to produce local maximum frequent itemssets*/

for (i = 1; i <= n; i++)

/*gaining global frequent items*/

{Scanning \( DB_i \) once;
computing local frequency of local items \(E_i\);
\(P_i\) sends \(E_i\) and local frequency of \(E_i\) to \(P_0\);
\}
\(P_0\) collects global frequent items \(E\) from \(E_i\);
\(E\) is sorted in the order of descending support count;
\(P_0\) sends \(E\) to other nodes \(P_i\); /*transmitting global frequent items to other nodes \(P_i\);*/
for(\(i=1; i<=n; i++\))
{creating the FP-tree \(i\);
\(\ast\)FP-tree \(i\) represents FP-tree of DB \(i\);
\(FM_i\) =DMFIA(FP-tree \(i\), \(min\_sup\));
}
step2/* \(P_0\) gets the union of all local maximum frequent itemsets */
for(\(i=1; i<=n; i++\))
\(P_i\) sends \(iFM_i\) to \(P_0\); /* \(iFM_i\) represents local frequent itemsets of \(P_i\);*/
\(P_0\) combines \(iFM_i\) and produces \(FM\); /* \(FM=\bigcup_{i=1}^{n}FM_i\);*/
step3/* \(P_0\) gets maximum frequent itemsets by the searching strategy of top-down and mapreduce */
\(FM=\emptyset\);
while \(FM\neq\emptyset\)
\{\(P_0\) confirms the largest size \(k\) of itemsets in \(FM\);
for all itemsets \(Q\in local frequent k\)-itemsets in \(FM\) /* \(P_0\) collects global frequency of \(Q\) from other nodes \(P_i\);*/
if \(Q\) are not the subset of any itemsets in \(FM\)
\{\(P_0\) broadcasts \(Q\);
\(P_i\) sends \(Q.si\) to \(P_0\); /* \(P_i\) computes local frequency \(Q.si\) of \(Q\) according to \(FP\)-tree */
\(Q.s=\sum_{i}Q.si\); /* \(Q.s\) represents global frequency of \(Q\);*/
if \(Q.s\geq\min\_sup\ast M\) /* \(Q\) are maximum frequent itemsets*/
\{ \(FM=FM\cup Q\);
\(P_0\) deletes \(Q\) and any nonempty subset of \(Q\) from \(FM\);
\}
else /* \(Q\) are not maximum frequent itemsets*/
\{ \(P_0\) deletes \(Q\) and any nonempty subset of \(Q\) from \(FM\);
for all item \(x\in Q\)
if \(Q\setminus\{x\}\) are not the subset of any itemsets in \(FM\)
\(FM=FM\cup \{Q\setminus\{x\}\}\);
\}
\}
\}

4. Experiments of MFIBD

This paper compares MFIBD with classical distributed mining algorithms CD and FDM. All tests are performed on 5 Lenovo PC as distributed nodes and 1 Dell workstation as center node. The experimental data comes from the sales data in August 2013 of a supermarket. The total number of tuples is about 20000. All programs are written in VC++ 6.0 and MPI.

Comparison experiment: It is a way of changing the minimum support threshold while adopting fixed number of nodes. MFIBD compares with CD and FDM in terms of database scanning times and runtime. The results are reported in Fig.1 and Fig.2.
The comparison experiment results indicate that under the same minimum support threshold, database scanning times and runtime of MFIBD decreases while comparing with CD and FDM.

5. Conclusion

MFIBD makes computer nodes calculate local frequent itemsets independently by DMFIA algorithm and mapreduce, then the center node exchanges data with other computer nodes and combines by the searching strategy of top-down. At last, maximum frequent itemsets are gained by mapreduce. Theoretical analysis and experimental results suggest that MFIBD is efficient.

6. Acknowledgment

This research is supported by the social science planning and cultivation project of chongqing under grant No.2014PY50 and the humanities and social science research project of chongqing municipal education commission under grant No. 15SKG131. This research is supported by the fundamental and advanced research projects of chongqing under grant No. CSTC2013JCYJA40039 and the national statistical science research project under grant No. 2015LZ22.

7. References


