A Distributed Frequent Itemsets Mining Algorithm Using Sparse Boolean Matrix on Spark

Yonghong Luo[1], Zhifan Yang[1], Huike Shi[1], and Ying Zhang[1,2(✉)]

[1] College of Computer and Control Engineering, Nankai University, Tianjin, China
{luoyonghong,yangzhifan,shihuike,zhangying}@dbis.nankai.edu.cn
[2] College of Software, Nankai University, Tianjin, China

Abstract. Frequent itemsets mining is one of the most important aspects in data mining for finding interesting knowledge in a huge mass of data. However, traditional frequent itemsets mining algorithms are usually data-intensive and computing-intensive. Take Apriori algorithm, a well-known algorithm in finding frequent itemsets for example, it needs to scan the dataset for many times and with the coming of big data era, it will also cost a lot of time over GB-level data. In order to solve those problems, researchers have made great efforts to improve Apriori algorithm based on distributed computing framework Hadoop or Spark. However, the existing parallel Apriori algorithms based on Hadoop or Spark are not efficient enough over GB-level data. In this paper, we proposed a distributed frequent itemsets mining algorithm by sparse boolean matrix on Spark (FISM). And experiments show FISM has better performance than all others existing parallel frequent itemsets mining algorithms and can also deal with GB-level data.

Keywords: Frequent itemsets mining · Apriori algorithm · Spark · Sparse matrix · FISM

1 Introduction

In order to solving the problem of association rules mining in the era of big data, many researchers have proposed lot of algorithms that using parallel computing technologies to deal with association rules mining. Yang X Y [4] proposed a distributed Apriori algorithm using Hadoop's MapReduce framework, however, this algorithm would cause a large number of I/O operations. Qiu H [3] proposed a parallel frequent itemsets mining algorithm with Spark called YAFIM (Yet Another Frequent Itemset Mining). And experiments show that, compared with the algorithms implemented with Hadoop, YAFIM achieved 18× speedup in average for various benchmarks [3]. But, YAFIM will also cost many hours when processing GB-level data. In the year of 2015, Zhang F proposed DPBM, a distributed matrix-based pruning algorithm based on Spark [1]. Experiments

F. Li et al. (Eds.): APWeb 2016, Part II, LNCS 9932, pp. 419–423, 2016.
DOI: 10.1007/978-3-319-45817-5_38

show that it is faster than YAFIM. But this algorithm may have a critical problem: if the total transactions are large enough, the computing speed will be very slow and not efficient.

In order to solve these problems, this paper proposed FISM, a new distributed frequent itemsets mining algorithm by using sparse boolean matrix on Spark. First, we use sparse matrix to replace the boolean-matrix of DPBM. Next, we also improved the generating process of candidate frequent itemsets. The experiments show that FISM outperforms DPBM, YAFIM, Parallel FP-Growth and other algorithms in both speed and scalability performance.

2 Distributed Frequent Itemsets Mining Algorithm by Using Sparse Boolean Matrix on Spark (FISM)

Assume that there are n transactions in the dataset D, and we call it $T = T_1$, $T_2, \cdots, T_n$. Next, assume that there are m items in the dataset called $I = I_1, I_2, \cdots, I_m$. The following are main steps of FISM:

First scan the dataset and for every item I_i ($\mathrm{i} = 1, 2, 3, \cdots, \mathrm{m}$), get the vector V_i ($b_{i1}, b_{i2}, b_{i3}, \cdots, b_{in}$) where

$$b_{ij} = \begin{cases} 1 & I_i \in T_j \\ 0 & I_i \notin T_j \end{cases}$$

($\mathrm{j} = 1, 2, 3, \cdots, \mathrm{n}$; $\mathrm{i} = 1, 2, 3, \cdots, \mathrm{m}$), and if $b_{ij} = 0$ then transaction T_j does not contain item I_i. There are total m vectors which build up the boolean matrix B. The number of columns of boolean matrix B is n and the number of rows of boolean matrix B is m. Sometimes, the number of transactions are very large, so the boolean matrix B will be implemented by sparse matrix. Figure 1 shows the actual data structure of B. For example, the i-th member of vectors is a list $(4, 7, 68, 6457, \cdots)$, so it means that transaction $4, 7, 68$ and 6457 all contain Item I_i and other transactions do not contain Item I_i. According to boolean matrix B, we can easily get the support number of Item I_i .

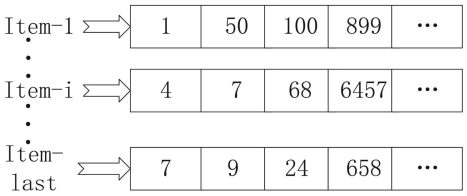


Fig. 1. Sparse Boolean matrix B

Definition 1. The "AND" result of one itemset E is defined as follows:

Suppose that E consists of h items, they are I_1, I_2, I_3, ..., I_h, and suppose that these items' corresponding rows in sparse boolean matrix B are V_1, V_2, V_3, ..., V_h, and suppose that $V_{result} = V_1$ & V_2 & V_3, ..., & V_h. Define that V_{result} is the "AND" result of E. The symbol "&" means "AND" operation between two vectors.

It is easy to generate (k + 1)-candidate frequent itemsets by k-frequent itemsets. After we get all k-frequent itemsets, for every k-frequent itemset, we stored its "AND" result (see Definition 1) in a list.Obviously, every (k + 1)-candidate frequent itemset consists of a new item I_{new} and a k-frequent itemset called $itemset_{old}$, and the "AND" result of $itemset_{old}$ is called V_{old}. Then V_{old} will be reused in the "AND" operation of V_{old} and I_{new}'s corresponding row of sparse boolean matrix B. In this way, we needn't to do k+1 times "AND" operations for the confirming of every one of (k + 2)-frequent itemset. We just do one time of "AND" operation of V_{old} and I_{new}'s boolean vector. Especially, we implemented the "AND" operation with multithreading technology.

In general,traditional frequent itemsets finding algorithms will scan the dataset for many times and this cause large amount of I/O operations while FISM only needs to scan dataset for once and use sparse matrix to accelerate the procedure of frequent itemsets finding. In theoretically, FISM outperforms traditional Apriori algorithm by up to one order of magnitude. And the experiments will confirm it.

3 Experiments Results

In this section, we compared FISM with MRApriori [4], YAFIM [3], DPBM [1] and PFP-growth (parallel FP-Growth [2] algorithm implemented by Spark team) to evaluate its performance. We implemented all the algorithms in Spark1.5.0. All the datasets are stored in HDFS and the cluster consists of 4 nodes and each node has 4 Intel Xeon cores with 2.60 GHz, 22.5 GB memory and 1 TB disk. The running system is CentOS6.5 and the version JDK is 1.7. Last but not least, **the correctness of all the algorithms mentioned above are exactly same**. Table 1 are characteristics of the datasets. Table 2 are results of experiments.

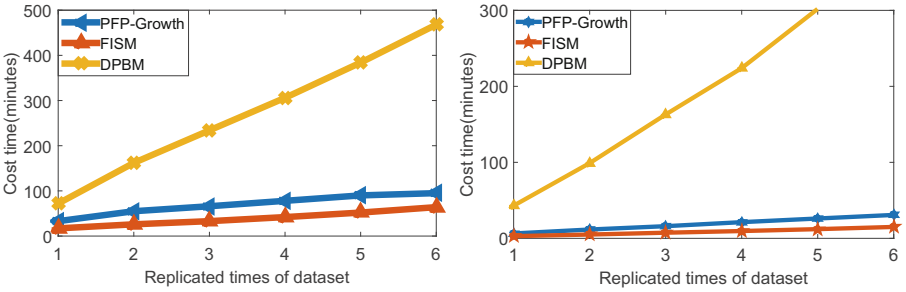
We can see that in every repetition and dataset, FISM outperforms all the others algorithms. For all the datasets, FISM is 1.8× faster than PFP-growth, 20× faster than YAFIM and 10× faster than DPBM.

Table 1. Detail properties of datasets

Dataset	Number of items	Number of transactions	Size
T10I4D100K	870	100000	3.9 MB
T40I10D100K	1000	100000	14.8 MB
Webdocs	1000	1692300	1.37 GB

Table 2. Experiments results

Dataset	YAFIM	MRApriori	DPBM	FP_Growth	FISM
T10I4D100K sup = 0.01	340 s	1920 s	75 s	32 s	15 s
T40I10D100K sup = 0.01	167 min	>12 h	43 min	6.6 min	2.9 min
Webdocs sup = 0.2	Out of memory	Out of memory	Out of memory	7.0 min	5.1 min



(a) T10I4D100K Sup=0.01 (b) T40I10D100K Sup=0.01

Fig. 2. The sizeup performance of each algorithms

Figure 2 shows the sizeup performance of all algorithms. The x-axis shows the number of replicated times of dataset. FISM is always faster than any other algorithms. MRApriori and YAFIM are not included in Fig. 2 because they spend more than 24 h. At the same time, we can see that with the increasement of dataset, the FISM 'cost time are increasing in a nearly linear way.

4 Conclusions

In order to accelerate frequent itemsets mining in big data era, this paper proposed FISM, a new distributed sparse boolean-matrix based frequent itemsets mining algorithm with Spark. It generate a sparse boolean matrix which shows whether one item is included in one transaction or not, then use the matrix to get all frequent itemsets. In this way, we only need to scan the dataset once. The experiments show that FISM is 1.8× faster than PFP-growth, 20× faster than YAFIM and 10× faster than DPBM. In addition, the FISM also has a better performance in scalability.

Acknowledgements. This work is partially supported by National 863 Program of China under Grant No. 2015AA015401, as well as the Research Foundation of Ministry and China Mobile under Grant No. MCM20150507. This work is also partially supported by Tianjin Municipal Science and Technology Commission under Grant No. 13ZCZDGX01098 and No. 16JCQNJC00500.

References

1. Gui, F., Ma, Y., Zhang, F., Liu, M., Li, F., Shen, W., Bai, H.: A distributed frequent itemset mining algorithm based on Spark. In: 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 271–275. IEEE (2015)
2. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD Record, vol. 29, pp. 1–12. ACM (2000)
3. Qiu, H., Gu, R., Yuan, C., Huang, Y.: YAFIM: a parallel frequent itemset mining algorithm with Spark. In: 2014 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 1664–1671. IEEE (2014)
4. Yang, X.Y., Liu, Z., Fu, Y.: Mapreduce as a programming model for association rules algorithm on Hadoop. In: 2010 3rd International Conference on Information Sciences and Interaction Sciences (ICIS), pp. 99–102. IEEE (2010)