An Incremental Association Rule Algorithm Based on MapReduce

Yong Liu\textsuperscript{1,2}, Jiecai Zheng\textsuperscript{1} and Xueqing Li\textsuperscript{1}

\textsuperscript{1}School of Computer Science and Technology, Shandong University, Jinan, China.
\textsuperscript{2}Department of Computer Engineering, Changji University, Changji, China.
Email: liuyong@sdu.edu.cn

Abstract. Association rules are one of the important methods of Data Mining, but the traditional association rule algorithm requires multiple scans of the database, with high I/O overhead and failure to handle node failure and load balancing. MapReduce is a popular distributed parallel computing model with good scalability and automatic load balancing and has been widely used. The existing algorithm for mining parallel association rules based on MapReduce computing model improves the computational efficiency. However, existing studies have ignored the characteristics of data increments. This paper will combine MapReduce and incremental data, and then develop and design incremental association rules based on MapReduce and combine the mathematical theory to design the mining algorithm. The effectiveness of the algorithm is verified through experiments.

1. Introduction
The rapid development of information in recent years has resulted in tremendous reforms in the old technologies. In particular, existing enterprises have gradually accumulated a great deal of data after informatization. However, with the advent of the cloud era, a large number of scholars began to focus on the relevant characteristics of these data to find out the relationship between data and promote the Make better decisions or increase the sales of your products [1]. The Market Basket Method is a famous example in the field of Data Mining [2][3][4]. Using well-designed algorithms to analyze supermarket business data, the researchers found items that are usually undetectable, enabling the store to design special shelves for related products, allowing the company's operations to be significantly improved.

Apriori [5] and FP-Growth [6] are well-known association algorithms. However, with the rapid development of information technology, these two methods are faced with bottlenecks such as memory shortage, processing efficiency, and low scalability due to a large amount of data in the era of Big Data. Even if the memory space is sufficient for use, processing a large amount of data will cause the algorithm to perform inefficiently and make the calculation result lose its effectiveness and lack of correctness. In the era of Big Data, the speed of data growth is unimaginable. If the algorithm's instruction cycle cannot be expanded in time, these algorithms cannot be applied to the computation of huge amounts of information. Therefore, many scholars proposed improvements for Apriori and FP-Growth. Wang et al. used parallel computing thinking and proposed the parFIM algorithm to solve the problem of memory shortage and low computational efficiency [7]. Although it can handle a large amount of data, it does not consider the dynamic configuration of the memory, resulting in the memory cannot be fully utilized. However, ZHAO et al. proposed improve Apriori algorithm [8], which uses Boolean Matrix to save transaction records. Although it can effectively reduce the memory...
usage, it still generates insufficient memory in the Big Data environment with a data size larger than Petabyte (PB). The problem.

Therefore, under the circumstance that the processing speed of the algorithm cannot adapt to the growth of data processing, the Apache Foundation proposed the MapReduce technology application based on distributed computing [9]. The use of multiple computer resources for parallel distribution processing, the huge data needs to be processed into a number of sub-tasks, the sub-tasks parallel distribution processing and then integrated, making the processing speed can be improved. Recent studies have begun to design the MapReduce framework for Apriori and FP-Growth. However, algorithms designed in this parallel distributed framework must prioritize data dependencies and cut the algorithm structure so that the algorithm can perform multiple subtasks. Perform calculations and finally integrate the results.

The main purpose of this paper is to distribute and cut internal tasks of FP-Growth algorithm so that it can cooperate with Hadoop MapReduce framework, which makes the algorithm avoid excessive memory usage and at the same time accelerates the processing speed of the algorithm.

2. Related Background Knowledge

2.1. Association Rules Classic Algorithm

The classic algorithms of association rule mining mainly include Apriori, FP-Growth and Eclat [10]. These three algorithms are still the basic algorithm of association rule mining.

The Apriori algorithm was proposed by Agrawal et al. in 1994[5]. Its main task is to search for frequent itemsets. The mining prioritization algorithm finds the hidden knowledge in the empirical data through the iterative process between layers and uses the output of the previous stage. Large sets of projects generate the next itemsets. The Apriori algorithm must rescans the database every time it checks whether a candidate item is a frequent item, resulting in a large amount of time wasted. Even if later scholars improve Apriori, their effectiveness is limited. In addition, many scholars have attempted to propose improved methods for data structure models. Among them, the FP-Growth algorithm is most famous. The algorithm improves the efficiency of mining. Although FP-Growth uses a special data structure for high compression volume reduction of the original transaction history record, in the Big Data era, even if the compression is still not able to adapt to such a large amount of data. Due to the excessive use of memory, the computing task is halted. In addition to this, while performing computational tasks, the database also has a considerable amount of new data to write at the same time. For these subsequently added data, the FP-Growth algorithm does not perform any processing and can only wait for the completion of the operation. Then re-add new data to re-calculate, so did not consider the newly added data, the final calculation results cannot show real-time analysis results, leading to inaccurate final results, which is FP-Growth faced in the Big Data operation challenge.

As discussed above, FP-Growth has the problems of insufficient memory, low efficiency, and low scalability in a stand-alone environment. To solve these problems, cloud computing technology brings the solution of MapReduce framework. However, not all algorithms are suitable for this framework. In order to parallelize computing tasks. If there is a strong dependency between data, the operation cannot be decomposed into multiple sub-operations for parallel decentralized operations. In addition to data dependencies, it is also necessary to consider how to decompose the operation structure of the algorithm and determine which frame in MapReduce to perform the operation of the task, and to comply with its paired <key, value> mode for data transfer. Therefore, not all algorithms can easily cooperate with the framework of MapReduce. Although it can bring more efficient operation modes, the existing single-machine calculation rules need proper design before parallel distributed computing.

2.2. MapReduce Framework

The MapReduce Framework is a programming model that supports parallel programming. It can implement distributed parallel computing of Big Data and has good scalability and fault tolerance.

Makanju A. et al. adopted the MapReduce framework and designed the MapReduce framework for the famous Apriori and FP-Growth association rule algorithms [11]. The research framework is shown
in Figure 1. First, use MapReduce to perform distributed operations, sharding the data in HDFS (Hadoop Distributed File System), and then send the sharded database to each Mapper by default. The Mapper will receive some of the original data. According to Apriori original operation mode, continuous iterations are used to generate high-order frequent item sets. Therefore, at the beginning of the algorithm, a \( k \) value is set, and \( k \) represents the number of iterations. The Mapper will then perform operations on the data and use the \(<\text{Tid}, \text{list}>\) previously entered in the Mapper to generate \( k \)-frequent itemsets. After Combiner operation, these data output from the Mapper are collectively rearranged using the same \(<\text{Set}_k, \text{value}>\). The Partioner uses \( \text{Hash}(\text{key}) \mod R \) operations to distribute the data and distribute the data to the Reducer. Then the \( k \) value will be calculated upwards in a cumulative manner. The \( k \)-Frequent itemset produced by the \( k \) iteration will generate the \( k+1 \) Frequent itemset, and the iterative calculation will be performed in the same way, until Apriori can no longer produce higher order frequent itemsets.

![Figure 1](image_url)  
**Figure 1.** Design of association algorithm based on MapReduce framework.

### 3. Association Rules Algorithm Design

This paper proposes to use MapReduce framework to make the algorithm can use multiple computer parallel operations to accelerate the computational efficiency and avoid the problem of insufficient memory. In order to minimize the total time spent on the overall operation, we defined Formula 1-2. Among them, \( T_m \) indicates the operation time of the Mapper, \( T_r \) indicates the operation time of the Reducer, \( D \) indicates the size of the data to be processed, \( B \) indicates the size of the Block, \( \lambda \) is the time required to allocate the data to be processed to the Mapper operation, and \( T_l \) indicates the operation total time spent.

\[
\lambda = \sum_{b=1}^{n} b D / B, \quad b = 1, 2, ..., n \text{ Block}
\]

\[
\min T_l = T_m + T_r + \lambda
\]

This paper uses the core theory of FP-Growth and Apriori, using Apriori object composition features and with FP-growth only need to scan the characteristics of the database once [12] [13], develops an association rule mining on the MapReduce framework, and can cope with new data continuously. The algorithm, we call it Parallel Incremental Association Rules Mining (PIARMA), the operation of the algorithm shown in Figure 2.
The algorithm is mainly divided into two parts: the first part is PIARMA phase, and the parallel association rule mining operation is performed through the MapReduce framework. The second part is the Incremental phase, and the parallel association rule mining of new data is performed through the MapReduce framework. PIARMA algorithm mainly includes the following three steps.

3.1. **Step1: 1-Frequent Itemsets Operations**

According to the traditional FP-Growth algorithm, it is necessary to perform the 1-Frequent itemsets operation in the first stage to establish the FP-Tree. A MapReduce process must go through this stage, as shown in Figure 3. After reading the data in the database, the Mapper will send its assigned data to the reducer in pairs of <key, value> parameters. For example, the information data is ACD. Mapper first splits the data into three separate items A, C, and D, and outputs <A, 1><C, 1><D, 1>, the meaning is that A, C, and D appear once in the data, and finally the Reducer performs the integration of the processed data and calculates the frequent items that appear in each item.

Using MapReduce in this section can effectively reduce the processing time and speed up the algorithm instruction cycle and performance. After calculating the 1-Frequent itemsets, frequent item screening will be performed because the FP-Growth algorithm is mainly aimed at mining the relevance in the data. If the number of occurrences of this item is sparse, it means that the item may lack common association with other items. Therefore, it is necessary to set a threshold value and filter it and delete the 1-Frequent itemsets that are lower than the threshold value to obtain the final frequent itemsets. Then the data in the database will be sorted in the descending order with reference to the 1-Frequent itemsets, and finally the *F_List* that continues to be used in Step2 will be generated.

**Figure 2.** PIARMA algorithm flow chart.

| TID | Item |
|-----|------|
| t1  | ABC  |
| t2  | CDE  |
| t3  | ABCE |
| t4  | BC   |

**Figure 3.** 1-Frequent itemsets and operations.
3.2. Step2: Combined Association rules

The traditional FP-growth uses the path in the tree structure to mine the association rules and uses the recursive method to perform the final object association mining. Therefore, while the FP-Growth algorithm is running, a mining tree is built in the memory in advance. However, this is the reason why the single-machine FP-Growth cannot handle a large amount of data. Therefore, the method proposed in this dissertation does not establish a mining tree, because in the MapReduce environment, there must be no dependencies among the data. In this way, the data and the algorithm can be separated and the task can be cut into sub-task parallel operations. Accelerate the purpose of the operation.

We have found that FP-Growth and Apriori both use the concept of combinatorial mathematics in the data mining process. For example, Apriori uses $k$-frequent itemsets to generate $k+1$-frequent itemsets in a combinatorial operation. In this process, the concept of permutation and combination is adopted. FP-Growth also produces the conditional pattern base generated by Data Mining at the end. The same applies to the concept of combinatorial mathematics.

### Table 1. Transaction record mode decomposition.

| Original | 2-frequent pattern | 3-frequent pattern | 4-frequent pattern |
|----------|--------------------|--------------------|--------------------|
| ABCD     | AB                 | ABC                | ABCD               |
| AC       | ABD                |                    |                    |
| AD       | ACD                |                    |                    |
| BC       | BCD                |                    |                    |

We also use the concept of combinatorial mathematics, do not rely on other data dependencies, based on a single transaction record for association mining. As shown in Table 1. After the Mapper reads the database to which HDFS is assigned, according to the default line-by-line format, after reading the TID, the Mapper first reads the $F_List$ in Step 1, and reads the TID according to the contents of the $F_List$. Sorting is performed to obtain the transaction record after the collation, and the disassembled records will be transmitted to the Reducer.

3.3. Step3: Merge Phase

In this phase, Reducer will receive the data that was decomposed by Mapper. Local frequent itemsets returned from each Mapper will be accumulated in the Reducer using WordCount. The use of WordCount allows MapReduce to use the original Combiner Function to transfer data to the Reducer in a nearly balanced allocation for unified integration, and finally produce a complete set of frequent items.

4. Experiments Analysis

4.1. Experimental Environment

In This paper presents experimental tests for Speed up, Size up, and incremental data processing for PIARMA and Parallel FP-Growth (PFP). A small clustered environment for experimentation, including 1-8 Clusters Nodes. The data used in the experiment is a set of webdocs transaction data provided by FIMI Repository (Frequent Itemset Mining Implementations Repository). The file size is 1.4 GB and contains 1.69 million transaction sets, of which the longest transaction set contains about 70,000 entries. In order to facilitate the experiment, based on the original file, the file was divided into 100 MB, 1 GB, and 1.4 GB partitions.

4.2. Speed Up Test

In To test the effectiveness of the algorithm in the Hadoop cluster, we performed a Speed up test. Speed up performs experiments for clusters with different number of nodes and tests the efficiency of the algorithm when increasing the number of nodes in the Cluster.

In the experiment, we fixed the amount of data to 1GB and used the change in the number of Nodes to calculate the effect. The experimental results are shown in Figure 4. The X axis represents the
number of cluster nodes used, and the Y axis represents the multiplier of the lifting effect. The acceleration effect using PIARMA increased by 5 times relative to 1 node in 8 nodes, and only 1.5 times in PFP in the same environment. PIARMA has a high degree of combinatorial complexity in single-machine environments and has poorer computational performance than PFP. However, as the number of nodes increases, tasks can be properly allocated to multiple nodes for computation, which can bring significant acceleration effects.

![Figure 4. Speed up experiment results.](image)

![Figure 5. Size up experiment results.](image)

### 4.3. Size Up Test

In this experiment, the amount of data used was 100MB to 1.4GB. The experimental results are shown in Figure 5. The X axis represents the amount of data used, and the Y axis represents the processing time of the algorithm. An interesting point to be observed in this experiment is that IPRAC performs worse than traditional PFP when the data volume is too small, but in the case of increasing data volume, the calculation efficiency can surpass the traditional PFP. This is a very interesting finding. According to our analysis of the algorithm, we found that this phenomenon should be caused by the uneven distribution of work load between Mapper and Reducer.

### 5. Conclusion and Outlook

This paper designs a MapReduce association rule algorithm based on permutation and combination, and it is measured by MapReduce platform. The concept of parallel distributed processing makes complex permutations and combinations run in the cloud computing environment. Experimental results show that this method is simple and effective. At the same time, we have also found that the complexity of using the permutation combination is too high, and in order to improve the efficiency of
the algorithm, we will also study the field of mathematics, and further study how to reduce the complexity of permutation and combinations in an image manner, and further strengthen Algorithm performance. In addition, the combination of the number of Mapper and Reducer, as well as the distribution of tasks, is also the main content of future research.

6. Acknowledgments
I should like to express my deepest gratitude to all those whose kindness and advice have made this work possible. This work is supported by the Key Research and Development Program of Shandong Province (2015GGX101009).

7. References
[1] Stanislaw Kozielski, Mrozek D, Pawel Kasprowski, et al. Beyond Databases, Architectures and Structures. Advanced Technologies for Data Mining and Knowledge Discovery [M]. Springer International Publishing, 2016.
[2] Srikant R, Agrawal R. Mining generalized association rules [J]. Future Generation Computer Systems, 2015, 13(s 2–3):161-180.
[3] Hanirex D K, Kaliyamurthie K P, Kumaravel A. Analysis of improved tdr algorithm for mining frequent itemsets using dengue virus type 1 dataset: A combined approach [J]. Indian Journal of Science & Technology, 2015, 6(2): B288-B295.
[4] Vo B, Le T, Coenen F, et al. Mining frequent itemsets using the N-list and subsume concepts [J]. International Journal of Machine Learning & Cybernetics, 2016, 7(2): 253-265.
[5] Agrawal, Rakesh, Srikant, et al. Fast algorithms for mining association rules[J] International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc. 1994:487-499.
[6] Han J, Pei J, Yin Y, et al. Mining Frequent Patterns without Candidate Generation: A Frequent- Pattern Tree Approach [J]. Data Min. knowl. disc, 2004, 8(1):53-87.
[7] Wang Y, Cao K. A Proactive Complex Event Processing Method Based on Parallel Markov Decision Processes[C] International Conference on Web-Age Information Management. Springer, Cham, 2014:237-241.
[8] ZHAO Yue, REN Yong-gong, LIU Yang. Improve Apriori Algorithm and Its Application Based on MapReduce [J]. Computer Science, 2017, 44(6): 250 - 254.
[9] Vaidya M. MapReduce: a flexible Data Processing Tool [J]. Communications of the Acm, 2010, 53(1):72-77.
[10] Heaton J. Comparing Dataset Characteristics that Favor the Apriori, Eclat or FP-Growth Frequent Itemset Mining Algorithms [J]. 2017.
[11] Makanju A, Farzanyar Z, An A, et al. Deep parallelization of parallel FP-growth using parent-child MapReduce[C] IEEE International Conference on Big Data. IEEE, 2017:1422-1431.
[12] Shen Yan, Zhu Yuquan, Liu Chunhua. Incremental FP_GROWTH Algorithm Based on Disk-resident 1-itemsets Counting [J]. Journal of Computer Reserch and Development, 2015, 52(3):569-578.
[13] Dharmaraajan K, Dorairangaswamy M A. Analysis of FP-growth and Apriori algorithms on pattern discovery from weblog data[C] IEEE International Conference on Advances in Computer Applications. IEEE, 2017.