

# An Improved Parallel Collaborative Filtering Algorithm based on Hadoop

Baojun Fu\*

*Institute of Computer Science and Information Engineering, Harbin Normal University, Harbin, 150025, China*

---

## Abstract

The existed parallel collaborative filtering algorithm based on co-occurrence matrix (CMCF) consumes a lot of time in the construction of co-occurrence matrixes and calculation of matrix multiplication. It also ignores the role of neighboring users, so it will influence the accuracy of recommendation. In order to solve this problem, this paper proposes the improved parallel collaborative filtering algorithm (IPCF) and its implementation on spark. The experimental results show that the improved parallel collaborative filtering algorithm in this paper has better running efficiency and higher recommendation accuracy.

**Keywords:** Hadoop; Spark; data mining; parallel collaborative filtering

(Submitted on December 19, 2017; Revised on January 22, 2018; Accepted on February 17, 2018)

© 2018 Totem Publisher, Inc. All rights reserved.

---

## 1. Introduction

The emergence of cloud computing enables research on big data mining to become effective. When the stand-alone device is unable to handle big data, it takes advantage of cluster parallel computing to process data on multiple machines simultaneously, greatly improving the data processing efficiency. In addition, with the increase in the number of cluster nodes, the computing speed will be correspondingly accelerated. To take advantage of the cloud platform's dynamic resource allocation and parallel computing capabilities, the primary problem to be solved is transformation of the traditional data mining algorithm into a parallel algorithm so that these algorithms can be effectively transplanted to the cloud platform. Therefore, research on algorithm parallelization in big data mining is of practical significance [1,14]. In the field of big data mining, many foreign researchers have not only aimed to improve the original algorithm under the serial of stand-alone devices, but more importantly, they have also begun to study the parallelization of data mining algorithms in a distributed cluster environment so that it can run in the distributed environment. Some results have been achieved. For example, the BIRCH algorithm [13] proposed by R. Ramakrishnan et al. and the DBSCAN algorithm [3] proposed by M. Ester et al. are effective improvements in the traditional clustering analysis algorithm under the big data environment. Sean Chester and Jeff Crow adopted the Map Reduce parallel programming model proposed by Google to carry out parallel improvement in the FP\_Growth algorithm, and they conducted research on problems and strategies in mining and processing. Good improvements are made in the implementation efficiency [2,6,8,9].

With the advent of the big data era, how to mine effective information from mass data and how to make it create values are all problems to be solved. In the traditional data mining algorithm [10,11,12], they were all based on the stand-alone device, but they could not do anything in the face of big data. Based on the research of the Hadoop and Spark platform and its combination with the HDFS and Spark parallel programming models, this paper improves the traditional data mining algorithm in parallel so that it is able to transplant to the cloud platform, which can effectively solve the problem of big data mining.

---

\* Corresponding author.

E-mail address: [baojunfufu@126.com](mailto:baojunfufu@126.com)

## 2. The process of collaborative filtering algorithm

By calculating the similarity between users, the neighboring user will be generated, and the target users will be recommended to the target users based on the items rated by the neighboring user. The steps of the algorithm are as follows:

- All users' ratings on all items are built into a  $m \times n$  matrix
- Based on the similarity calculation algorithm, the correlation between target users and other users is calculated.
- Sorting by the size of the similarity degree, taking the largest K user as the target user's neighboring user
- According to the predictive calculation algorithm, the expected value of the target user to the neighboring user is calculated. The first N items with the maximum expected value are recommended to the target user.

The algorithm pseudo code is shown below.

---

Algorithm 1 Collaborative filtering algorithm pseudo code.

---

Input: rating matrix R, target user u

Output: collection  $P_u$  of recommended items generated for the target user u

```

1.  $P_u = \emptyset$ 
2.  $W_{sim} = \emptyset$ 
3.  $K_{neighbor} = \emptyset$ 
4.  $I_{neighbor} = \emptyset$ 
5. For (i=1; i<=m; i++){
    If (i!=u){
         $W_i = sim(u, i)$ 
         $W_{sim} \leftarrow W_i$ 
    }
}
6.  $K_{neighbor} \leftarrow TopK(W_{sim})$ 
7.  $I_{neighbor} \leftarrow items\ in\ K_{neighbor}$ 
8.  $P_u \leftarrow TopN\ in\ predict(I_{neighbor})$ 

```

### 2.1. Rating matrix

The database will record all users' ratings of all items; in general, the two-dimensional  $m \times n$  rating matrix R is composed of "user - item". It is shown in Table 1. M is the number of users, N is the number of items. Each element  $R_{ij}$  in the matrix is the user i's rating for item j.  $R_{ij} \in \{0, 1\}$  is like and dislike.  $R_{ij} \in \{1-5\}$  shows that the higher the rating is, the higher the preference is.

Table 1. Rating matrix

	$I_1$	$I_2$	$I_3$	$I_4$
$U_1$	3	2	5	4
$U_2$	5	3		1
$U_3$	4	4		
$U_4$	3	1	3	3
$U_5$	1		2	4

## 2.2. Similarity calculation method

As seen from the algorithm flow, the algorithm focuses on how to calculate the similarity between users according to the scoring matrix. Similarity calculation determines the accuracy of the recommendation directly. The commonly similarity calculation method is as follows.

### 1. Pearson correlation coefficient

The Pearson correlation coefficient [5] is a method proposed by Pearson to describe the correlation of two vectors; the larger the value is, the greater the correlation is. The smaller the value is, the smaller the correlation is. Take the user-based Collaborative filtering (CF) as an example. Set vector  $I(u) = (R_{u1}, R_{u2}, R_{u3} \dots R_{un})$ , which represents the user  $u$ 's rating set for all items, i.e. the rows corresponding to the user  $u$  in the rating matrix. The Pearson correlation coefficient calculates the similarity between user  $u$  and user  $v$  as follows. It is shown in Equation (1).

$$sim(u, v) = \frac{\sum_{i \in I(u) \cap I(v)} (R_{ui} - \bar{R}_u)(R_{vi} - \bar{R}_v)}{\sqrt{\sum_{i \in I(u) \cap I(v)} (R_{ui} - \bar{R}_u)^2} \sqrt{\sum_{i \in I(u) \cap I(v)} (R_{vi} - \bar{R}_v)^2}} \quad (1)$$

### 2. Cosine similarity

In the field of text mining, cosine similarity [4] is often used to calculate the similarity between texts. The degree of correlation is described by calculating the angle between two vectors. The smaller the angle is, the more similar the two vectors are. The bigger the angle is, the more dissimilar the two vectors are. In the scoring matrix, each row represents the rating vector of each user, and the similarity between user  $u$  and user  $v$  is calculated by the cosine similarity. The Equation (2) is as follows.

$$sim(u, v) = \cos(u, v) = \frac{u \times v}{|u| \times |v|} = \frac{\sum_{i=1}^n R_{ui} R_{vi}}{\sqrt{\sum_{i=1}^n R_{ui}^2} \sqrt{\sum_{i=1}^n R_{vi}^2}} \quad (2)$$

### 3. Modified cosine similarity

In the computation of cosine similarity, there is no consideration of the difference between the degrees of user rating. For example, some users believe that 5 points is a high score and 3 points is low. Some users believe 3 points is high. Therefore, some people have proposed a modified cosine similarity, adding the average value of the user's rating to effectively reduce the impact of the difference in the degree of user rating. The similarity between user  $u$  and user  $v$  is calculated by the modified cosine similarity. It is shown in Equation (3).

$$sim(u, v) = \frac{\sum_{i=1}^n (R_{ui} - \bar{R}_u)(R_{vi} - \bar{R}_v)}{\sqrt{\sum_{i=1}^n (R_{ui} - \bar{R}_u)^2} \sqrt{\sum_{i=1}^n (R_{vi} - \bar{R}_v)^2}} \quad (3)$$

## 2.3. Recommended prediction

The similarity set  $W_{sim}$  of the target user  $u$  and other users is calculated by the similarity algorithm. In  $W_{sim}$ , the  $K$  user with the largest similarity is found as the neighboring user set  $K_{neighbor}$  of the target user  $u$ . The Top  $K$  can be calculated from the collection by a fast sorting algorithm or a heap sort algorithm. Predicting user  $u$ 's rating for item  $i$  can be calculated by the item of the neighbor user. The calculation Equation (4) is as follows:

$$P(u, i) = \frac{\sum_{v \in K_{neighbor} \cap U(i)} sim(u, v) \times R_{vi}}{|\sum_{v \in K_{neighbor} \cap U(i)} sim(u, v)|} \quad (4)$$

Where  $P(u, i)$  represents the prediction rating of user  $u$  for item  $i$ ,  $U(i)$  represents the set of user set for item  $i$ . It

represents the similarity between user  $u$  and user  $v$ . In Equation (3), it does not take into account the difference between the degrees of user rating. Some users believe that 5 points is a high score. Some users believe that 3 points is a high score. Therefore, the user's rating mean is added to effectively reduce the influence of user rating difference. Equation (5) makes the following improvements:

$$P(u, i) = \bar{R}_u + \frac{\sum_{v \in K_{neighbor} \cap U(i)} sim(u, v) \times (R_{vi} - \bar{R}_v)}{|\sum_{v \in K_{neighbor} \cap U(i)} sim(u, v)|} \quad (5)$$

### 3. Existing parallel collaborative filtering algorithm based on co-occurrence matrix(CMCF)

#### 3.1. Algorithm parallelization idea.

In the co-occurrence matrix, row  $i$  and column  $j$  represent item  $i$  and item  $j$  and appear together in a user. The co-occurrence matrix represents the degree of correlation between the two items. The more time the two items are together, the higher the correlation is. It is show in Table 2.

Table 2. The co-occurrence matrix

	$I_1$	$I_2$	$I_3$	$I_4$	$I_5$	$I_6$
$I_1$	5	3	4	4	2	2
$I_2$	3	3	3	2	1	1
$I_3$	4	3	4	3	1	2
$I_4$	3	2	3	4	2	2
$I_5$	2	1	1	2	2	1
$I_6$	1	0	0	1	1	0

#### 3.2. Problems existing in the algorithm

The existing parallel collaborative filtering algorithm can solve the problem of big data set, yet the following problems exist:

Firstly, when there are many projects, the co-occurrence matrix will be very large, which will consume a lot of time in building the co-occurrence matrix. When calculating the target users' prediction rating vector, the co-occurrence matrix multiplying the user rating vector will consume a lot of time. In the distributed computing, the matrix multiplication efficiency is poor. Secondly, the collaborative filtering algorithm based on the co-occurrence matrix has ignored the role of neighboring users. Depending on the frequency of project co-occurrence, there may be more correlation between the relationships. They are not based on the similarity calculation method, which to a certain extent has affected the accuracy of recommendation. To handle the above two problems, this paper proposes an improved parallel collaborative algorithm, achieving specific implementation under the Spark distributed computing framework. In view of the above two problems, this paper proposes an improved parallel collaborative filtering algorithm and implementation based on Spark distributed computing framework.

#### 3.3. Improved parallel collaborative filtering algorithm(IPCF)

Based on the co-occurrence matrix's collaborative filtering algorithm parallelization problems, the improved parallel collaborative filtering algorithm is introduced. It is named the IPCF algorithm in this study. The improved parallel algorithm mainly consists of three steps: generating the rating matrix, getting the neighboring user, and generating the recommendation. The steps to get the neighboring users are the main improved points. In the past, the computation of Map Reduce and Spark was to first divide the data into multiple small blocks and store them on different computing nodes. Each node performed computation for different data blocks respectively. In order to meet the computation of large-scale data, it is not directly expressed in the form of a two-dimensional matrix when storing the rating matrix. Instead, it is represented by a triple <user ID, item ID, rating>, which stands for the user ID, item ID and rating. Each user's rating for each item is represented by a triple and stored in the file for each triple, and all ratings are stored to one or more files in HDFS.

##### (1) Generating rating matrix

We use Map Reduce or Spark to calculate the collection of triples. In the map phase, with each row data <user ID, item ID, rating> is value input, user ID is key, <user ID, rating> is value output. In the reduce phase, the same user ID corresponds

to  $\langle \text{item ID}, \text{rating} \rangle$  will be distributed to the same reducer. Then, according to user ID, merge value to get KV output:  $\langle \text{user ID}, \text{list}(\text{item ID}, \text{rating}) \rangle$ . This is the same for each row of data in the rating matrix. Due to the parallel computation of multiple nodes, the output results of reducer on different nodes will be stored on different nodes, resulting in the user's rating matrix.

### (2) Access to neighboring user

The similarity calculation method is used to find the K neighboring user of the target user. For example, the Pearson and cosine similarity calculation method. The K minimum heap algorithm is used to find the maximum value of K similarity. Before the start of the map phase, the researcher first constructs the minimum heap in the size of K to store K largest neighboring users with the largest similarity. It then uses the output generated by the previous job as input in the map, i.e., the user vector  $\langle \text{user ID}, \text{list}(\text{item ID}, \text{rating}) \rangle$  as input value, computes the similarity between each user and the target user in succession, compares and adjusts them in the minimum heap. In the minimum heap, K neighboring users have the greatest similarity with the target user should be maintained. After the end of the map phase, the Key Value output is obtained:  $\langle \text{user ID}, \text{Heap}(\text{user ID}, \text{sim})K \rangle$ . In the reduce phase, all the map outputs are merged, that is, to merge the minimum heap of all map outputs, take advantage of the merge comparison algorithm, find the leading K users in the maximum value from all  $\text{Heap}(\text{user ID}, \text{sim})K$ , and get the final Key Value output:  $\langle \text{user ID}, \text{list}(\text{user ID}, \text{sim}) \rangle$ . The list (user ID, sim) is the neighboring users and similarity of the target user.

### (3) Recommendation

By using the recommendation prediction algorithm, we calculated the expected value of the target users to the neighboring user, and took the first N items with the highest expected value as the recommendation. It needs to use the output of the first two jobs at the same time as input. Before the map phase, first read the output of the second job  $\langle \text{user ID}, \text{list}(\text{user ID}, \text{sim}) \rangle$  to memory, which is to put K neighboring user and their similarity into the memory. However, in the map calculation, the output  $\langle \text{user ID}, \text{list}(\text{item ID}, \text{rating}) \rangle$  is used as the value input in the first job. If the user ID is a neighboring user belonging to the target user, the output  $\langle \text{user ID}, \text{list}(\text{item ID}, \text{rating}), \text{sim} \rangle$ . Otherwise filter out the user rating vector. In the reduction phase, the output of all maps is integrated, and all item rating vectors of the K neighboring users are merged. It uses the recommendation and prediction algorithm  $P(u, i)$  and the minimum heap algorithm to find the top N items with the maximum prediction rating. The N items are the best recommended items for the target user. From the above three steps, it can be seen that the improved IPCF algorithm is easy to understand and easy to implement. Avoid building a large co-occurrence matrix and complex matrix multiplication operation, and find the neighboring users of the target users by computing. Find the best recommendation item in the neighboring user rating item, which conforms to the thought and process of the traditional collaborative filtering algorithm.

## 4. Implementation of improved IPCF algorithm on Spark

Each job completes a part of the work by converting the algorithm into a series of computing pipelining that is made up of Spark job. The process is as follows:

### 4.1. Generating rating matrix

$\langle \text{user ID}, \text{item ID}, \text{rating} \rangle$  used as input (With  $\langle u, i, r \rangle$  said). Using the characteristics of the Spark model, the user rating vector is generated. It is shown in Equation (6).

$$U = [u, (i_1, r_1), (i_2, r_2), \dots, (i_m, r_m)] \quad (6)$$

The text File interface is provided by the Spark Context, and the input File stored in HDFS is converted to the RDD of Spark's initial calculation. Then, "User RatingMapToPair" and "UserVectorMapToPair" are designed to complete the generation of User Vector RDD.

#### 1. UserRatingMapToPair

---

UserRatingMapToPair main steps

---

**Input:**  $\langle (u, i, r) \rangle$ ,  $\langle u, i, r \rangle$  represents three tuples

**Output:**  $\langle u, (i, r) \rangle$ ,  $u$  represents the user ID,  $(i, r)$  represents the object of the item ID and the value of the rating

(1). Read the truples  $\langle u, i, r \rangle$  and get the user ID, item ID and rating;

(2). Use  $u$  as key,  $(i, r)$  forms a two tuple object as a value;

(3). Return tuple2  $\langle (u, i, r) \rangle$  key value pair;

(4). End.

## 2. UserVectorMapToPair

### UserVectorMapToPair main steps

**Input:**  $\langle (u, \text{Iterable}(i, r)) \rangle$ ,  $u$  represents the user ID,  $\text{Iterable}(i, r)$  represents two tuples

**Output:**  $\langle (u, \text{userVector}(i, r)) \rangle$ ,  $u$  represents the user ID,  $\text{userVector}(i, r)$  represents the vector of the item ID and the value of the rating

- (1). Define a user vector  $\text{userVector}$  ;
- (2). Remove  $(i, r)$  objects from the  $\text{Iterable}(i, r)$  iterators and add them to  $\text{userVector}$  ;
- (3).  $u$  as key,  $\text{userVector}$  as value;
- (4). return to tuple2  $\langle (u, \text{userVector}(i, r)) \rangle$  key value pair.

### 4.2. Access to neighbor user

User “VectorMapToPair” as output,  $\langle (u, \text{userVector}(i, r)) \rangle$  as input. Using Spark to provide “flat MapToPair”, “reduce By Key”, “mapToPair” to compute neighbor user. Design “NeighborMapToPair” and “Neighbor ReduceByKey” to generate target user’s neighbor user.

## 1. NeighborMapToPair

### NeighborMapToPair main steps

**Input:**  $\langle (u, \text{userVector}(i, r)) \rangle, k$

**Output:**  $\langle (u, \text{Heap}(u, \text{sim}))k \rangle$

- (1). In the initialization of the task, the smallest heap of the size of  $K$  is built in memory;
- (2). Loading the target user's rating vector into memory;
- (3).  $\langle (u, \text{userVector}(i, r)) \rangle$  as input, the similarity of each user and the target user is calculated by using the similarity calculation Equation;
- (4). The similarity and minimum heap are compared and adjusted;
- (5). The target user  $u$  is used as key before the end of the task. The smallest heap is generated as value;
- (6). Output  $\langle (u, \text{Heap}(u, \text{sim}))k \rangle$ .

## 2. NeighborReduceByKey

### NeighborReduceByKey main steps

**Input:** Iterate  $\langle (u, \text{Heap}(u, \text{sim}))k \rangle, k$

**Output:**  $\langle (u, \text{list}(u, \text{sim})) \rangle$

- (1) Define a new minimum heap size of  $K$ ;
- (2) Each local minimum heap is extracted from the Iterate  $\langle (u, \text{Heap}(u, \text{sim}))k \rangle$  iterator;
- (3) Each item in the local minimum heap is compared and adjusted to the new  $K$  minimum heap;
- (4) After the comparison, the global front  $K$  neighbor is obtained;
- (5) New  $K$  minimum heap output to list  $\text{list}(u, \text{sim})$ ;
- (6) Output tuple2  $\langle (u, \text{list}(u, \text{sim})) \rangle$  key value pair.

### 4.3. Recommendation

After getting the neighboring user of the target user, it finds the recommended item in the neighboring user's rating item. It needs to use the output of the first two jobs as input. First, the rating vectors belonging to the neighboring user are selected from all user rating vectors, and then the prediction value of the target user to the item is calculated in the neighbor rating vector. Design “RecommendedMap” and “RecommendedReduce”.

## 1. RecommendedMap

### RecommendedMap main steps

**Input:**  $\langle \text{userID}, \text{list}(\text{userID}, \text{sim}) \rangle, \langle \text{userID}, \text{list}(\text{itemID}, \text{rating}) \rangle$

**Output:**  $\langle \text{userID}, \text{list}(\text{itemID}, \text{rating}), \text{sim} \rangle$

- (1).  $\langle \text{userID}, \text{list}(\text{userID}, \text{sim}) \rangle$  is loaded into memory when the task is initialized;
- (2). Read the user rating vector input, that is  $\langle \text{userID}, \text{list}(\text{itemID}, \text{rating}) \rangle$ ;

- (3). In the user rating vector, if the *userID* is included in the neighbor user set, then the user's rating vector and similarity output are obtained. Otherwise, no processing is made to the user rating vector;
- (4). *userID* as key,  $\langle \text{list}(\text{itemID}, \text{rating}), \text{sim} \rangle$  as value
- (5). Output  $\text{tuple2} \langle (\text{userID}, \text{list}(\text{itemID}, \text{rating})), \text{sim} \rangle$  key value pair.

## 2. RecommendedReduce

RecommendedReduce main steps

**Input:**  $\langle \text{userID}, \text{list}(\text{itemID}, \text{rating}), \text{sim} \rangle$

**Output:**  $\langle u, \text{recommendedItems} \rangle$

- (1) Take out the rating list from  $\langle \text{userID}, \text{list}(\text{itemID}, \text{rating}), \text{sim} \rangle$ ;
- (2) Each item and its rating value in the rating list are taken out in order, and the target user is calculated by the recommendation prediction algorithm;
- (3) By the minimum heap or quicksort algorithm, the top N items in all the predicted rating values were obtained
- (4) The largest first N Items are encapsulated in the *recommendedItems*;
- (5) Output  $\text{tuple2} \langle u, \text{recommendedItems} \rangle$  key value pair.

## 5. Experiments and results

### 5.1. Data sources

The experimental data comes from the Movie Lens data set provided by Group Lens. Movielens, established by Group Lens [7] is for non-commercial purpose of studying the recommended technology. Movie Lens contains multiple data sets of different sizes that can predict the recommended algorithm by testing Movie Lens's different data set. Movie Lens data set contains 71,567 users and 10,681 movies, as well as each user's rating for the movies, with a rating of 100,000 entries. The rating entry format is one entry per line: user ID, movie ID, and rating value. The rating range is 1-5. It is shown in Table 3.

Table 3. Movie Lens data format

Field	Meaning	type	Range
User ID	USER ID	Int	1-4156
Item ID	movie ID	Int	1-1068
Rating	rating value	Float	1.0-5.0

The size of this experiment is the Movie Lens data set of rating record, 100W, 200W, 300W, and... 1000W. The data set is respectively applied to the implementation of the Spark-based collaborative filtering algorithm. Calculate the computation time for each input data set, and the results are presented by a chart. The experimental environment uses common machines to build Hadoop and Spark clusters, including a master node and several slave nodes. The configuration difference of each node is very small. The environment is 2 core processors, 4G memory, 300G hard disk, K-MB network card, Ubuntu13.04 64bit operating system.

### 5.2. Running time

The collaborative filtering algorithm implemented in this paper is not for the improvement of the algorithm itself, but to make the algorithm to adapt to the distributed computing based on the cloud platform, especially under the premise of big data. Therefore, the running time can measure whether the improved parallel collaborative filtering algorithm is improved in terms of operational efficiency. In this experiment, an improved parallel collaborative filtering algorithm based on the stand-alone device and a distributed cluster composed by several nodes is respectively run on a different data set of varied sizes. It runs for multiple times over each combination, and the average value is taken as the final result. The specific operation time of each job can be seen in the Spark application web page. It is shown in Figure 1.

As you can see from Figure 1, for data set with the same size, the time required for algorithm running decreases with the number of nodes. In the case of a small data set, the time difference is small; as the data set increases, the time difference is obvious. This shows the ability of parallel computation.

### 5.3. Speed-up

Speed-up on a single processor system and is the ratio between running elapsed time and multiple processors elapsed time. It is used to measure the parallel system processing or parallelization of the program indicators. The higher the acceleration ratio is, the better the performance of the parallel is. It is shown in Equation (7).

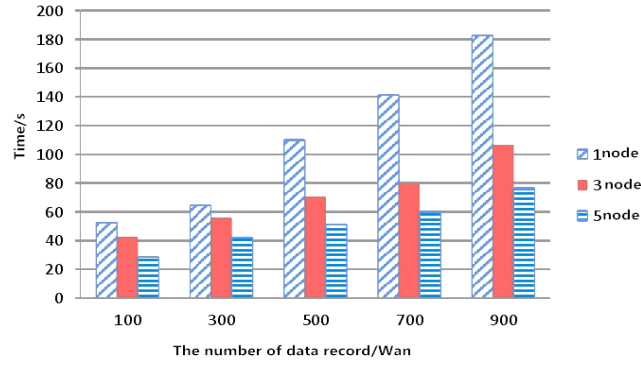


Figure 1. Improved IPCF algorithm runtime diagram

$$S_n = \frac{T_1}{T_n} \quad (7)$$

In order to test the parallel processing performance of the improved parallel collaborative filtering algorithm, different data sets  $D_1(100W)$ ,  $D_2(400W)$ ,  $D_3(700W)$ ,  $D_4(1000W)$  are used in this experiment. It runs on the cluster of single and different nodes, calculates the time required, and runs the average to get the final result. Then, we calculate the Speed-up in all cases. It is shown in Figure 2.

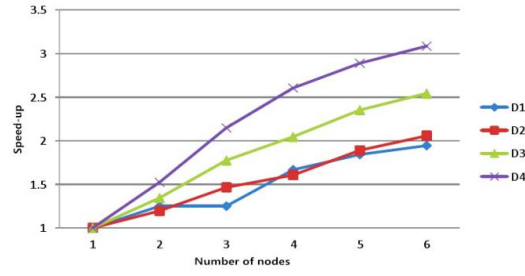


Figure 2. Speed-up diagram

As you can see from Figure 2, the improved parallel collaborative filtering algorithm has an obvious speed-up ratio effect. With the same number of nodes, the larger the data set is, the larger the speed-up ratio is, which indicates the ability of Spark to handle big data. Moreover, as the number of nodes increases, the speed-up ratio will be increased correspondingly, and it will tend to be slow. This is because each node needs the network for communication; as the number of nodes increases, the network overhead also increases in computation. The experimental data itself is not large enough and the cluster is also small (subject to experimental conditions), which cannot fully reflect the actual ability to handle big data in parallel. Therefore, the speed-up ratio tends to be slow. In this experiment, the improved parallel collaborative filtering algorithm IPCF is compared with the existing CMCF. Under the same data set and the same experimental environment, the improved parallel collaborative filtering algorithm IPCF has a higher time running efficiency than that of CMCF; the time difference is more obvious when the data set is larger. It is shown in Figure 3-Figure 6.

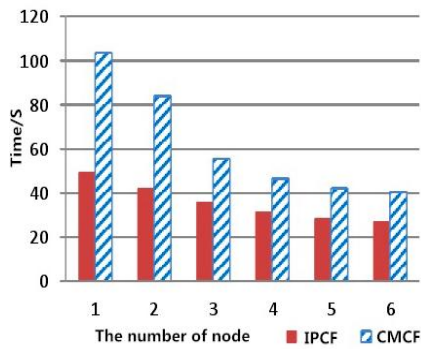


Figure 3. time comparison of Data set D1

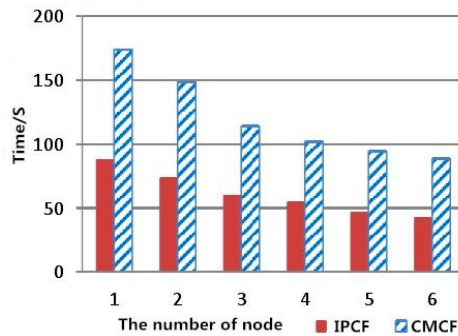


Figure 4. time comparison of Data set D2



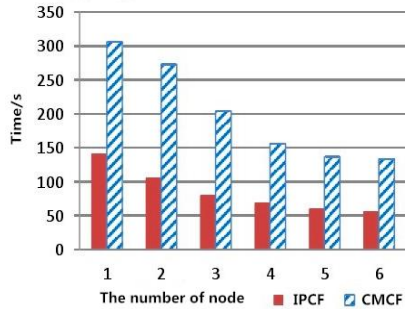


Figure 5. time comparison of Data set D3

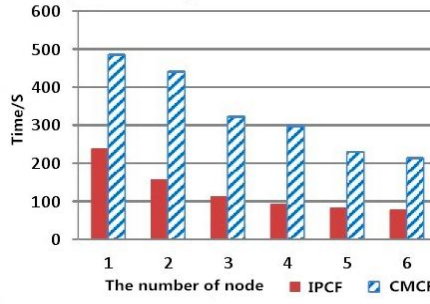


Figure 6. time comparison of Data set D4

#### 5.4. Recommendation accuracy comparison

MAE (mean absolute deviation) and RSME (root-mean-square error) methods are often used in evaluating the quality of the recommended algorithm. MAE is the more widely used method for evaluating the recommended algorithm, which reflects the discretization degree of rating data by computing MAE between the predictive value and the actual value. If  $I_a$  is used to represent the target user  $a$ , there is a set of items with both a predictive rating and  $a$  actual rating. MAE calculation Equation (8) is:

$$MAE = \frac{\sum_{i \in I_a} |P_{a,i} - r_{a,i}|}{N} \quad (8)$$

In the computation result, the smaller the MAE value is, the higher the accuracy of the algorithm is. The experimenter uses the data set provided by Movie Lens and randomly divides the data set into training set and testing set. The training set occupies 80% of the total data and the testing set accounts for 20%. In this experiment, the improved parallel collaborative filtering algorithm IPCF is compared with the existing CMCF based on the co-occurrence matrix, and their MAE values are compared under the same data set and the same experimental environment. The user ratings are predicted on the testing set. Selection of different data sets and determination of the MAE value is shown in Figure 7. As you can see from Figure 7, as the size of data sampled increases, MAE tends to decrease, i.e., the accuracy of the algorithm is higher. When the data size exceeds 5 million, MAE obtained by IPCF algorithm running tends to be stable. At the same time, it can be seen that MAE obtained by IPCF algorithm is generally smaller than that by the CMCF algorithm, indicating that the improved IPCF algorithm has better recommendation accuracy than that of the CMCF algorithm. Therefore, it can be concluded that the improved parallel collaborative filtering algorithm has higher recommendation accuracy than the parallel collaborative filtering algorithm based on the co-occurrence matrix. It owns the improved running efficiency and is better suited to computations under the big data set.

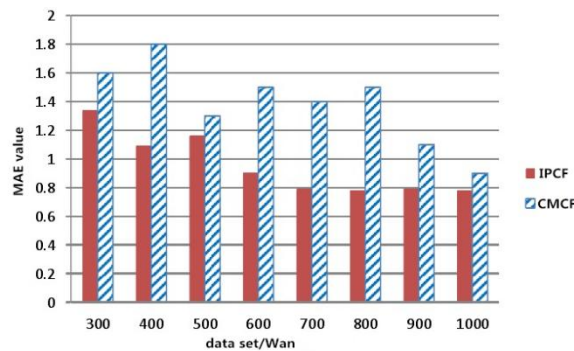


Figure 7. MAE comparison diagram

## 6. Conclusions

The traditional collaborative filtering algorithm and the existing parallelism based on the co-occurrence matrix were analyzed, the improved parallel collaborative filtering algorithm was proposed, and the specific implementation of the improved parallel collaborative filtering algorithm on the Spark platform was elaborated. Experiments show that the improved parallel

collaborative filtering algorithm has higher recommendation accuracy and has good speed-up ratio; the larger the data set is, the better its advantages and scalability are. In this paper, some achievements have been made in the research on the parallelization of the traditional data mining algorithm, and the collaborative filtering algorithm has been improved in parallel. However, there are many algorithms in the field of data mining, and other algorithms and parallelization need to be further studied.

## References

1. G. Bart. "Memory Issues in Frequent Itemset Mining". *Proc of ACM Symposium on Applied Computing*, New York, NY: ACM, pp.530-534, 2004
2. C. Cheng, "Research on Cloud Platform Recommendation Algorithm", *Chongqing University of Technology*, 2014.
3. M. Ester, Hans-peter. Krieger, "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", *Proc of the Second International Conference on Knowledge Discovery and Data Mining*. Menlo Park, California: AAAI Press. pp. 226-231, 1996
4. S. Gill. "Introduction to Modern Information Retrieval", *Mc Graw-Hill*, New York, NY, USA, 1983.
5. L. Herlocker, "A Collaborative Filtering Algorithm and Evaluation Metric That Accurately Model the User Experience", in *Proceedings of 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Sheffield, UK, pp.329-336, 2004.
6. C. Li, "Recommendation Algorithm and Application of MapReduce Based on Hybrid", *Computer Technology and Development*, vol.26, no.4, pp. 74-77, 2016
7. "Movie Lens: Film Recommendations", [Http://movielens.umn.edu](http://movielens.umn.edu).
8. L. Qi, "Research on Collaborative Filtering Algorithm Based on MapReduce", *Taiyuan University of Technology*, 2014.
9. B. Tian, P. Hu. "Research on Collaborative Filtering Recommendation Algorithm Based on clustering," *Computer Engineering and Science*, vol.38, no. 8, pp. 1615-1624, 2016
10. Y. Wen, D. Wu, "Personalized Education Resources in The Spark Platform", *The Intelligent Computer and Application*, vol.7, no. 2, pp.25-30, 2017
11. M. Xu, H. Shen, "Spark Parallelization Based on object Collaborative Filtering Algorithm", *Computer Engineering and Design*, vol.38, no.7, pp.1817-1822, 2017
12. C. Zhang, "Research and Implementation of Hadoop Based Collaborative Filtering Algorithm", *Donghua University*, 2015.
13. T. Zhang, "An Efficient Data Clustering Method for Very Large Databases", *Proc of the 1996 ACM SIGMOD International Conference on Management of Data*. New York, NY: ACM, pp.103-114, 1996
14. W. Zhao, J. Li, "Hadoop Cloud Platform Based on User Collaborative Filtering Algorithm Research", *Computer Measurement and Control*, vol.23, no.6, pp.2082-2085, 2015

**Baojun Fu** received his B.S degree from Qiqihar University. He received his M.S degree from Harbin Engineering University. He is a lecturer at the Institute of Computer Science and Information Engineering of Harbin Normal University. His research interests include artificial intelligence and social network.