Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 5, May 2018, pp. 927-936
DOI: 10.23940/ijpe.18.05.p11.927936

Collaborative Filtering Recommendation Algorithm based on Cluster

Zhiyong Li*

Hunan Mass Media Vocational and Technical College, Changsha, 410100, China

Abstract

The traditional collaborative filtering recommendation method suffers from sparse datasets, cold starts, and efficiency problems. Furthermore, recommend accuracy decreases with an increase in the amount of data. Therefore, we improved the traditional collaborative filtering recommendation method by increasing the same rating between users when calculating their similarity and running it on a cluster. Because of the above actions, the collaborative filtering recommendation method obtains a better accuracy. Through experiments, we saw that the method we proposed has higher accuracy and efficiency compared to traditional collaborative filtering recommendation methods.

Keywords: big dataset; recommender system; collaborative filtering

(Submitted on February 13, 2018; Revised on March 22, 2018; Accepted on April 25, 2018)

1. Introduction

With the rapid development of the Internet, the number of users and recommended objects of the recommended system has grown exponentially, consuming a lot of time to implement these algorithms on a single-node machine. It fails to meet the computation needs of large databases [2,11,12]. The cost of computer hardware continues to decrease, leading to a constant advancement of computer cluster technology, and various platforms based on the computer cluster have been established. Therefore, it is essential to propose a recommendation algorithm that adapts to large databases by combining current computer cluster technology with a traditional collaborative filtering recommendation algorithm, which can also promote the wide application of collaborative filtering recommendation algorithms in large databases [4,5].

In the era of big data, there are massive amounts of users and items in the large e-commerce and/or social networking sites [1,6,8]. With this massive data in the recommended system, data sparseness has become a headache. Normally, the number of user rating items only accounts for a small proportion of the total number of items. With an increasing number of items and users, this proportion will get smaller and smaller, leading to sparse rating data. In this case, the common rating between users will be very small, causing a low recommended effect on the recommended system [9,10]. As a result, when introducing the calculation of item similarity and calculating similarity between users, this paper first calculates the user rating item, concentrates the items that are not rated simultaneously, and then calculates the similarity of two users according to the calculated rating value, thus, settling the data sparseness. At the same time, due to different rating habits, some users prefer high scores, whereas others prefer low scores. The same person rates the same item differently, yet the preference degree may be the same [3]. Therefore, this paper introduces the subjective rating normalization method, which recalculates the rating matrix and eliminates the error caused by a distinct user rating scale.

2. User based Collaborative Filtering Algorithm and its Improvement

2.1. Subjective Rating Normalization

Regardless of online e-commerce websites or news blog sites, different users show varied rating habits towards the same item.

* Corresponding author.
E-mail address: zhiyongli2017@126.com

Some users casually rate and give high scores to most items, whereas others hold relatively careful and meticulous attitudes and give low scores to their favorite items. It is thus obvious that different rating scales have a large impact on the calculation of similarity when there is a large difference in the ratings of the same item. Even if such an issue had been considered, it remains rough and inaccurate. This paper introduces the subjective rating normalization algorithm, which converts all scores of users to the same value coordinates system, normalizes the average score of rated users, and obtains the new user-item rating matrix, thereby eliminating the error arising from distinct rating scales. Subjective rating normalization algorithm is a new user-item rating matrix $A(m,n)$ that converts the user-item rating matrix $A^{'}(m,n)$ to a normalized user rating. The specific implementation process is as follows.

Step 1. According to the input rating matrix, calculate the average score $\overline{r}_u$ for each user u, as shown in Equation (1).

$$\overline{r}_u = \frac{\sum_{i=1}^{n} r_{u \to i}}{n} \tag{1}$$

Step 2. Reset the user rating interval and solve all the minimum and maximum sum of the means as the value of interval extremes. It is shown in Equation (2).

$$\left[\frac{\sum_{j=1}^{m} \min(r_{u_j \to i_1}, r_{u_j \to i_2, \ldots} r_{u_j \to i_n})}{m}, \frac{\sum_{j=1}^{m} \max(r_{u_j \to i_1}, r_{u_j \to i_2, \ldots} r_{u_j \to i_n})}{m} \right] \tag{2}$$

Step 3. Recalculate score $r^{'}_{u \to i}$ of the user u over the item i and produce a new user-item rating matrix $A^{'}(m,n)$.

2.2. User rating forecast

In the calculation of user similarity, the traditional collaborative filtering algorithm is commonly used for rating items on the collection of computing. According to the traditional similarity calculation equation, there will be two people with different results. So, through the introduction of the item similarity calculation, the prediction rating algorithm will increase the common rating between users and solve the data sparseness between users brought about by the decline in quality of such issues [7]. User rating prediction algorithm is based on user-item rating matrix $A(m,n)$, the user is not on the rating items calculated rating $p_{u \to i}$. The specific implementation process is as follows.

Step 1: Calculate the similarity between items based on the user's rating matrix. It is shown in Equation (3).

$$sim(i_m, i_n) = \frac{\sum_{u \in U_{i_m \cap i_n}} (r_{u \to i_m} - \overline{r}_{im})(r_{u \to i_n} - \overline{r}_{in})}{\sqrt{\sum_{u \in U_{i_m \cap i_n}} (r_{u \to i_m} - \overline{r}_{im})^2 (r_{u \to i_n} - \overline{r}_{in})^2}} \tag{3}$$

Step 2: Calculate the predicted score of the user u_i over the item $I_{i \cup j}$ according to the item similarity calculation using Equation (3) (assuming that the user u_i does not rate the item i_m, it is certain that u_j has rated the item i_m according to the union calculation rules). It is shown in Equation (4).

$$p_{u \to i_m = \overline{r}_{u_i}} + \frac{\sum_{i \in I} sim(i_m, i)(r_{u_i \to i} - \overline{r}_{u_i})}{\sum_{i \in I} sim(i_m, i)} \tag{4}$$

The above algorithm can be used to calculate the user's score of the non-scoring items. This method can reduce the influence of data sparsity on computing user similarity and improve the accuracy of the proposed algorithm.

2.3. Improved Collaborative Filtering Recommendation Algorithm

By adding the subjective rating normalization to the traditional collaborative filtering algorithm, this paper may reduce the impact of inconsistent rating criteria on recommended results. In addition, it can increase the common user rating score and reduce data sparseness when using the predicted rating calculation method and calculating the similarity between users. The implementation process of the improved collaborative filtering recommendation algorithm is as follows.

Step 1: According to the user item rating matrix, $A(m,n)$, calculate each user score between user-items. It is shown in Equation (5).

$$I_{i\cup j} = I_i \cup I_j \tag{5}$$

Step 2: Normalize score matrix based on the subjective score normalization algorithm. It is shown in Equation (6).

$$A(m,n) \to A^{'}(m,n) \tag{6}$$

Step 3: The similarity between users is calculated based on the user's item set $I_{i\cup j}$ and the normalized user item score matrix $A^{'}(m,n)$. It is shown in Equation (7).

$$sim(u_a,u_b) = \frac{\sum_{i\in I}(r_{u_a\to i} - \overline{r}_{u_a})(r_{u\to i} - \overline{r}_{u_b})}{\sqrt{\sum_{i\in I}(r_{u_a\to i} - \overline{r}_{u_a})^2 (r_{u_a\to i} - \overline{r}_{u_b})^2}} \tag{7}$$

Step 4: Assuming that the user u_a is the target user, the user i_m is to be predicted, according to the Equation of user similarity (7), the user u_a can predict the item i_m . It is shown in Equation (8).

$$p_{u_a\to i_m=}\overline{r}_{u_a} + \frac{\sum_{u\in U} sim(u_a,u)(r_{u\to im} - \overline{r}_{u_a})}{\sum_{u\in U} sim(u_a,u)} \tag{8}$$

3. Parallel implementation of collaborative filtering algorithm on Cluster

3.1. Algorithm design idea

The collaborative filtering recommendation algorithm based on cluster technology adopts the distributed computing matrix and distributed file system storage in order to solve the bottleneck performance of traditional recommendation algorithms in a single machine. The design of the algorithm is described as follows. First of all, the massive user rating data required by the recommended system are saved for the cluster distributed file system, which is completed by the Hadoop distributed file system. The essence is to cut some large files and save each storage child node by a certain sized block. Secondly, each computing task is partitioned into sub-tasks through Hadoop's Map Reduce distributed computing framework, which is passed to each sub-node of the computing cluster. The processing results are recursively returned after each sub-node completes the computing sub-tasks.

3.2. Map Reduce of Subjective Rating Normalization Algorithm

The subjective rating normalization algorithm is mainly responsible for converting the user-item rating matrix into a unified value coordinate system. The subjective rating normalization algorithm is divided into three steps concurrently. The first step is to solve the user-item rating matrix based on the user line. The second step is to solve the average maximum and minimum sum according to the user-item rating matrix. The third step is to solve the normalized user-item rating matrix according to the user-item rating matrix and the average maximum and minimum sum. The first step is to solve the user item rating matrix. The user-item scoring matrix algorithm flow chart is shown in Figure 1.

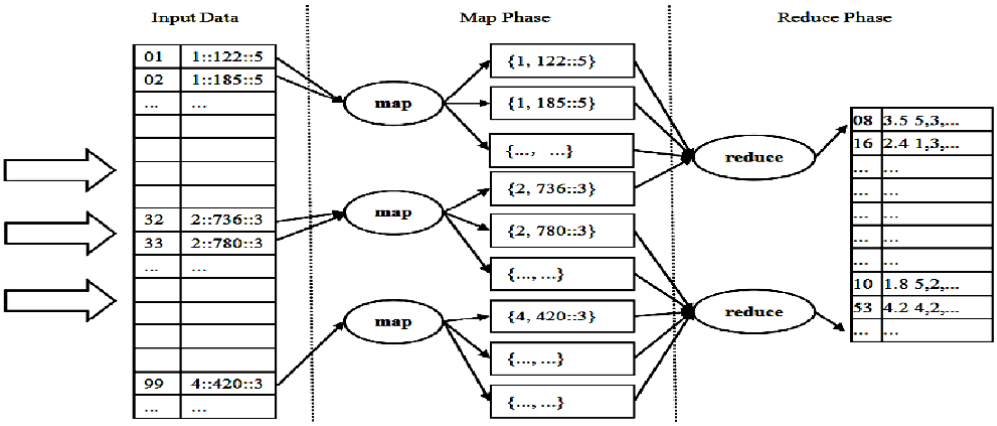

Figure 1. The process of user item scoring matrix based on MapReduce

Input Data: Input Splite under org.apache.hadoop.mapred is mainly used at this phase and the storage is achieved by line. The rating data file with the delimiter "::" is cut into blocks of the same size, and those blocks are distributed among each computing node. Generally, the default size of the default Input Split reading files is similar to the size of default blocks of Hadoop HDFS, and the sizes of default files may be adjusted by modifying the configuration file. Record Reader is responsible for loading the Input Slite data block and converting the data into a key-value pair suitable for Mapper reading. The files are read by line offset in files as the Key and the line contents as the Value. Map Phase: Map operation is achieved by implementing Mapper interface based on user-defined inherited Map Reduce Base at this phase. When Record Reader reads the data from the file in a specified manner, it will submit the file to the user-defined class by the key-value pair of {key, value}, which has a user-defined map method in the class. The Map phase mainly analyzes the file line saved in the format of the Text, i.e. the value. The string is split into a string array by "::", and the user ID is taken as the key at the Reduce phase. The item ID and the item scores will constitute a 2-tuple as the value at the Reduce phase. Reduce Phase: This phase is to read the output at the Map phase. The key of the Map phase is the user ID, and the value means the user rating over a certain item. Map Reduce framework will pass the key-value pair with the same Key to a Reduce for merging. Therefore, each user's rating information may be merged with this point to generate the user's rating line over the entire item set, and the average item score of the user will be solved. Eventually, the {key, value} pair with the user ID as the Key, the average score, and all item scores over the item set as the Value are saved to the HDFS file system.

The following is a pseudo code for the process, as follows.

```
  map(LongWritable key, Text value, OutputCollector<Text, Text> output)
{
1 extract userId, itemId and rating from value
2 output. collect("userId", "{itemId, rating}");
}
reduce(Text key, Iterator<Text> values,OutputCollector<Text, IntWritable> output,){
1. define an array to store rating
2. for({itemId, rating} in values){
  add rating to array
  }
3. sort the array
4. get the average rating from array
5. output. collect("userId", "average rating : rating array")
};
```

The above process is the way to get the user's score matrix according to the user's rating data. The score matrix is stored in a file of the HDFS distributed file system, and each line of the file contains three data contents, namely the user ID. The second step is to solve the max min plus average. The maximum and minimum average solution algorithm flow chart is shown in Figure 2.

Input Data: The file format and the first step of the file format are different. The user ID and user rating information for the entire item are set as lines, and we need to make the score information correspond to the item number. Insert a score and

an item that is not scored into the position of the item number on the serial number by inserting 0. Map Phase: This phase is mainly used to analyze the value of the user's evaluation of the entire project set branch. Then, find out the maximum score of the items and the smallest items, respectively, to {key, value} pairs of the situation passed to the Reduce stage, where the key is the string "Max" or "Min" and value corresponds to the key. If the key is "Max", the maximum value of the row is populated. If it is "Min", the minimum value of the row is filled. Reduce Phase: This phase is mainly based on the output of the Map phase, where the key has only two values: "Max" or "Min". The MapReduce framework will be thrown into a collection with the same key value. Therefore, all the elements in the set can be obtained directly, and then the average value is divided by the total number. The results obtained by {key and value} are in the form of a written HDFS file system. The key system is "Max" or "Min" and the "Max" value is the set of all values of the sum and average value. The "Min" in the collection of all the values and the average value are added.

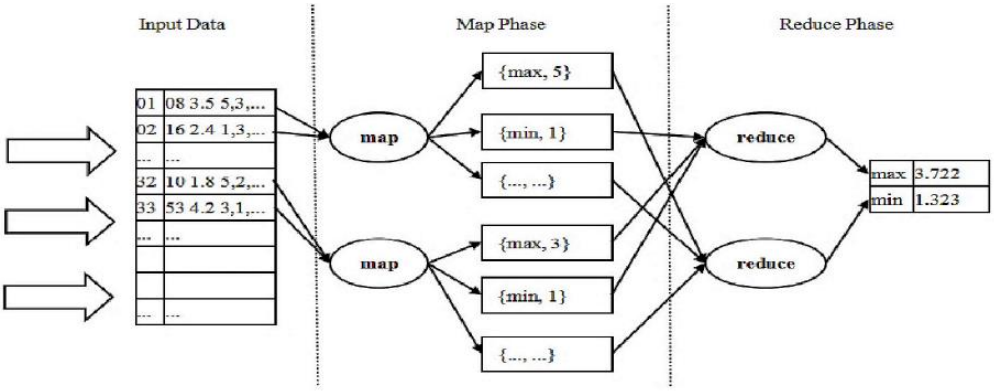

Figure 2. The maximum and minimum mean square sum solution of user scoring based on MapReduce

This process is based on the user score matrix in order to solve the maximum and minimum weighted score values. After the completion of the solution, it generates a "Max" and "Min" list of values and then takes the average value of the normalized algorithm to obtain the k value, which is stored in the HDFS distributed file system. The third step is to solve the normalized user item score matrix. The normalized user-item score matrix algorithm flow chart is shown in Figure 3.

Input Data: This phase is used to read the user-item rating matrix previously generated with the average score. The user's weighted average of the rated items will be used in the user rating normalization algorithm. Map Phase: This phase is to resolve the rating line of the user in the value over the entire item set, obtain the user average score of rated items, read the k value from the HDFS distributed file system, generate the new rating line by using other rated normalized scores according to the Equation, and pass the results to the Reduce phase in the form of a {key, value} pair. Among this, the key is the user ID and the value is generated for a new normalized user rating line. Reduce Phase: This stage is mainly used to write the {key, value} output from the Map phase to the HDFS distributed file system for the calling of later steps.

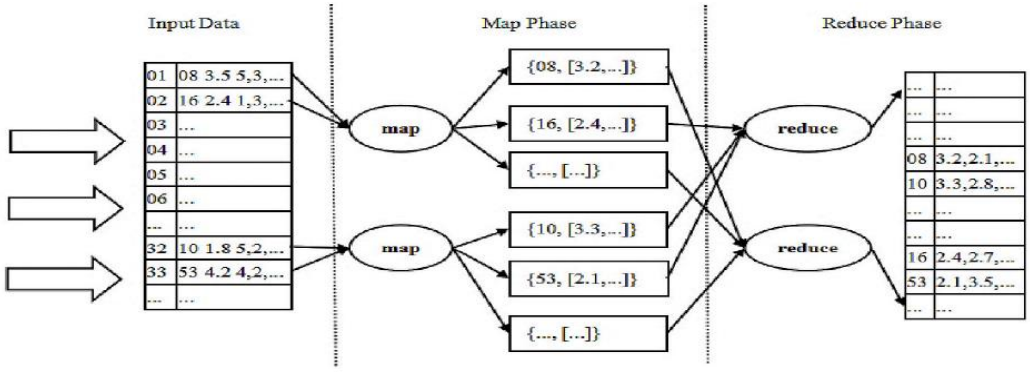

Figure 3. Normalized solution of user item scoring matrix based on MapReduce

The following is a pseudo code for the process, as follows.

```
map(LongWritable key, Text value,OutputCollector<Text, Text> output){
1. extract userId, average rating and user-item rating from value
2. read k value from HDFS file system
3. calculate the normalized rating for each user-item rating
```

```
4. output. collect("userId ", "normalized user-item rating");
}
reduce(Text key, Iterator<Text> values,OutputCollector<Text, IntWritable> output,){
1. for(rating in values){
2.output.collect(" userId", "normalized user-item rating");
  } };
```

The above procedure is based on the user item rating matrix used to solve the normalized user-item score matrix.

3.3. Map Reduce of Predicted Rating Algorithm

Predicted rating of the algorithm relies mainly on the calculation of item similarity, and the predicted rating of the user over similar items are calculated by the item similarity. Therefore, the algorithm is divided into two steps concurrently. The first step is to calculate the similarity between the various items, and the second step is to calculate the predicted rating of users over similar items. Item similarity calculation algorithm flow is shown in Figure 4.

Input Data: This phase still reads the normalized user-item rating matrix generated previously. Map phase: This phase is used to analyze the rating line of the user in the value over the entire item set, obtain the user rating information of all items, merge the simultaneously rated items, and pass the results in the form of a {key, value} pair to the Reduce phase. Reduce Phase: This phase calculates the item similarity. Map Reduce framework will throw the same key, i.e. the same value of {items ID， items ID}, into a set. Take all {item rating，item rating} pairs in the set and obtain the similarity between items according to Equation 5. After completion of the calculation, with {item ID, item ID} as the key and the similarity as the value, they are written into the HDFS distributed file system.

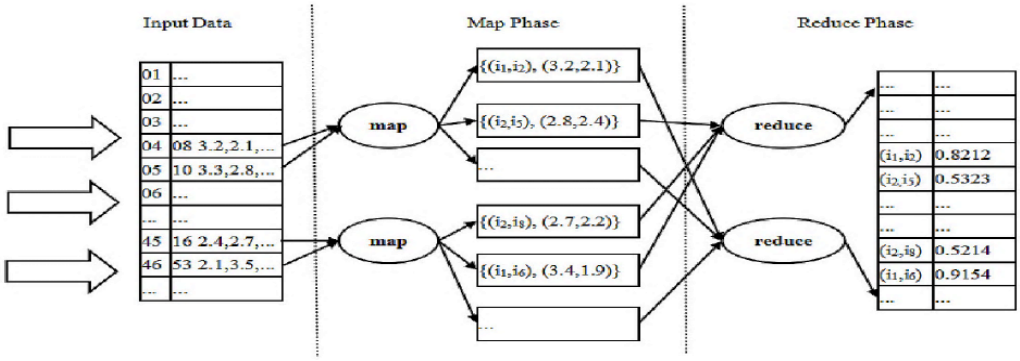

Figure 4. The process of item similarity based on MapReduce

The following is a pseudo code for the process, as follows.

```
map(LongWritable key, Text value,OutputCollector<Text, Text> output){
1. extract user-item rating and itemId from value to set S
2. for(itemId in S){
   //find another item with none zero rating
output.collect("itemId pair", "rating pair");
  }}
reduce(Text key, Iterator<Text> values, OutputCollector<Text, IntWritable> output,){
1. extract all rating pair from values
2. calculate the item similarity
3. output. collect(" itemId pair", "item similarity");
};
```

This process is based on the user-item score matrix to solve the item similarity. The second step is item similarity ranking. Item similarity ranking algorithm flow chart is shown in Figure 5.

Input Data: this stage is the first step that reads the item similarity file. Map Phase: this stage splits the analytic value, splits the string ID to get the item and item similarity, and then recombines them with the {key, value}, which will then be passed to the Reduce stage. Reduce Phase: This phase calculates the item similarity. Map Reduce framework will throw the same key, i.e., the same value of {items ID， items similarity}, into a set. Take all {item rating，item rating} pairs in the set

and obtain the similarity between items. After completion of the calculation, with {item ID, item similarity} as the key and the similarity as the value, they are written into the HDFS distributed file system.

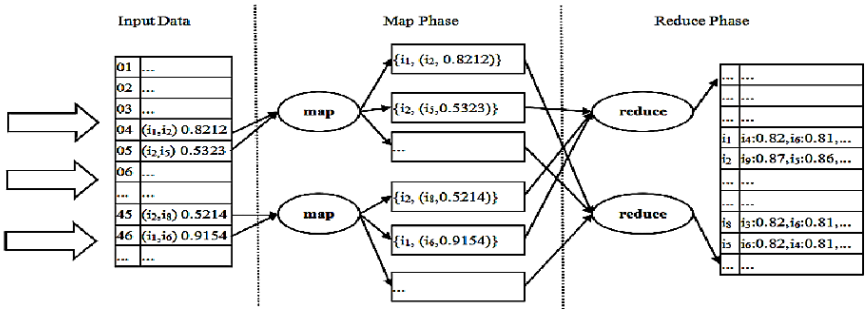

Figure 5. Item similarity ranking based on MapReduce

The following is a pseudo code for the process, as follows.

```
map(LongWritable key, Text value, OutputCollector<Text, Text> output){
1. extract itemId pair and item similarity from value
2. output. collect("first itemId", "(second itemId, item similarity)");
};
reduce(Text key, Iterator<Text> values, OutputCollector<Text, IntWritable> output,){
1. extract all item similarity from values to set
2. sort the item similarity set
3. output. collect(" itemId", "sorted item similarity set");
};
```

The predictive scoring algorithm flow chart is shown in Figure 6.

Input Data: this phase reads the normalized user item score matrix. Map Phase: this stage splits the string parsing value, ID access and each item score. If the score is 0, according to the similarity calculation of the item, the item of user rating prediction based on the {key, value} on the form will be passed to the Reduce stage, where the key is the user ID. Reduce Phase: this stage is used for the Map output of the key, where the key is the user ID and the value for the user is used to predict the score of all items not rated.

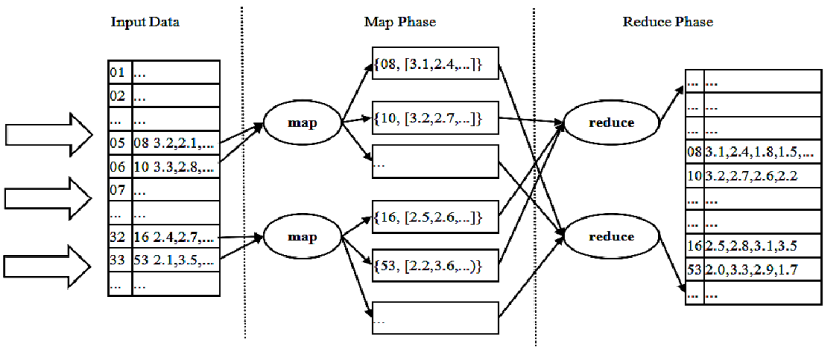

Figure 6. Prediction scoring process based on MapReduce

The following is a pseudo code for the process, as follows.

```
map(LongWritable key, Text value, OutputCollector<Text, Text> output){
1. extract userId and item rating from value
2. define a set.
3. if(user[i].rating == 0){
4. calculate prediction rating
5. set.add(itemId, prediction rating); }
6. output.collect("userId", "set");  }
reduce(Text key, Iterator<Text> values, OutputCollector<Text, IntWritable> output,){
1. extract all itemId, prediction from values
2.output. collect(" userId", [(itemId,prediction),...])};
```

4. EXPERIMENTAL ANALYSIS AND RESULTS

4.1. Construction and deployment of experimental environment

In this experiment, a total of three machines are used on the cluster platform, and the configuration of each machine is basically the same. The operating system is the 32 version of the ubuntu13.10 desktop, and the three machines each deployed 2 virtual machines. We ensure that the virtual machine can be interconnected with each other through a ping communication environment. Secondly, the host name and the static IP address are set for each machine, and the correspondence between the IP address of the machine and the host name is stored in the host file of each machine.

4.2. Experimental data set

In this paper, the data set of online video site MovieLens (http://movielens.umn.edu) is used as the experimental data set. The data set contains three compressed packets: ml-100k.zip, ml-1m.zip and ml-10m.zip. MovieLens data sets are stored in the user ID, project ID, score, timestamp, etc. The experiment was used in the ml-100 compression package u.data (1.88Mb), ml-1m compression bag. Ratings.dat (23.4Mb) and ml-10m.zip compressed package ratings.dat (252Mb) files are used as the training data set and test data set.

4.3. Experimental measurement standard

In order to measure the accuracy of the recommendation algorithm, this paper uses the average absolute deviation MAE as the measurement standard. Assume that the actual user rating set for the test set is $\{q_1, q_2, ..., q_N\}$, and the predicted user rating set is represented as $\{p_1, p_2, ..., p_N\}$, then the mean absolute deviation MAE is defined as Equation (9).

$$MAE = \frac{\sum_{i=1}^{N} | p_i - q_i |}{N} \tag{9}$$

At the same time, in order to measure the performance of the proposed algorithm, the concept of speedup is defined. The Equation for the acceleration ratio is defined as Equation (10).

$$R = \frac{T_s}{T_c} \tag{10}$$

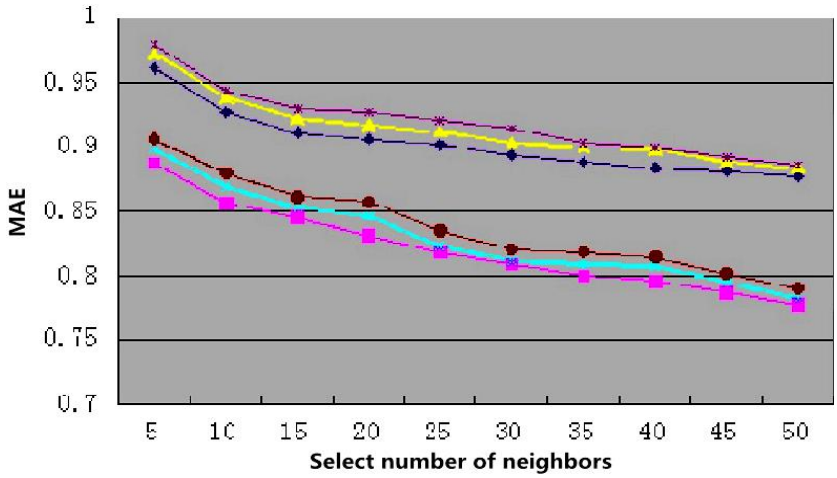

Figure 7. The effect of improved collaborative filtering recommendation algorithm

4.4. Experimental result analysis

4.4.1. Comparison of the accuracy of collaborative filtering recommendation algorithm before and after improvement

In order to verify the effectiveness of the proposed algorithm, the traditional collaborative filtering recommendation algorithm

based on user similarity and the improved collaborative filtering recommendation algorithm are compared, as shown in Figure 7. Experimental results show that the improved collaborative filtering recommendation algorithm outperforms the traditional collaborative recommendation algorithm based on user similarity in the same data set. In addition, even if the sparseness of the 1M and 10M data set is greater than 100K under different data sets, good recommended results can be obtained through the normalized user rating matrix and the increase in predicted rating of similar items. Therefore, the improved collaborative filtering algorithm can solve the data sparseness issue.

4.4.2. Speedup of parallel algorithm

The parallelization of the algorithm is mainly used to test the algorithm execution time with the number of changes in the node, as shown in Figure 8.

According to the experimental results, the speed-up ratio is inconsistent with the growth of the number of nodes because the pure increase in the number of nodes also consumes more computing resources required by communicating and scheduling the nodes. As a result, the number of nodes does not maintain a 1-to-1 growth with the speed-up ratio. Moreover, the greater the amount of data is, the more obvious the speed-up ratio effects are because it takes more time to undertake the computing task in the node with the increase in the amount of data and there is less time required for communication. Consequently, the speed-up ratio effect becomes more obvious with one increased node in the large database.

5. Conclusions

In the traditional collaborative filtering algorithm, the subjective score normalization and predictive scoring algorithm are integrated into the improved algorithm collaborative filtering algorithm. In this paper, we focus on the parallel algorithm of the subjective scoring normalization algorithm and predictive scoring algorithm. The performance of each algorithm and the efficiency of the proposed algorithm are analyzed by experiments, and the reasons are analyzed.

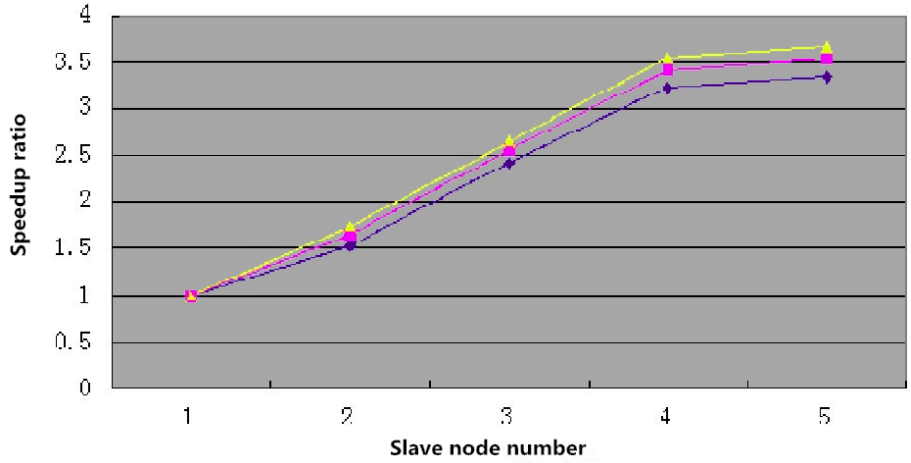

Figure 8. The algorithm execution time with the number of changes in the node

References

1. Yaya Au, Chaomu Yuan, "Improved Collaborative Filtering Recommendation Algorithm Based on Trust Degree", *The modern library and information technology*, vol.10, pp.49-53, 2011
2. Haitao Chen, Shanshan Song, "Recommendation Algorithm for Collaborative Filtering Based on User", *Intelligence theory and practice*, vol.2, pp.100-103, 2015
3. Shu Cheng, Lin Gui, "Subjective Rating Normalization Algorithm and Error Analysis", *Journal of Higher Correspondence Education (Natural Sciences)*, vol.21, no.5, pp.143-147, 2011
4. Yaoning Fang, Yunfei Guo, Lan. Hu, "An Improved Collaborative Filtering Recommendation Algorithm Based on Sigmoid Function", *Computer application research*, vol.5, pp. 1688-1691, 2013
5. Wangneng Li, "Research on Improved Collaborative Filtering Recommendation Algorithm Based on Hadoop", *Chongqing University*, 2015
6. Xudong Liu, Deren Chen, Huimin Wang, "An Improved Collaborative Filtering Algorithm", *Journal of Wuhan University of Technology (Information Science Edition)*, vol.4, pp.550-553, 2010
7. P. Melville, RJ. Mooney, R. Nagarajan, "Content-Boosted Collaborative Filtering for Improved Recommendations". In: Proc. of

the 18th National Conf. on Artificial Intelligence. Menlo Park: *American Association for Artificial Intelligence*, pp.187−192, 2002.
8. Hongchen Wu, Xinjun Wang, "An Improved Recommendation Algorithm Based on Collaborative Filtering and Partition Clustering", *Research and development of computer science*, vol. S3, pp. 205-212, 2011
9. Junbo Wang, "Collaborative Filtering Recommendation Algorithm and Its Improvement", *Chongqing University*, 2010
10. Fang Yang, Hong Pan, "An Improved Collaborative Filtering Recommendation Algorithm", *Journal of Hebei University of Technology*, vol.6, pp. 82-87, 2010
11. Lanping Ye, "An Improved Collaborative Filtering Recommendation Algorithm", *Anhui University*, 2016
12. Xu Zhang, "Improvement of Collaborative Filtering Recommendation Algorithm and Implementation of Distributed Computing", *Shandong University*, 2015

Zhiyong li received his M.S degree from Nation University of Defense Technology. He is a Lecturer in Hunan Mass Media Vocational and Technical College. His research interests include big data, internet of things.