International Journal of Modern Developments in Engineering and Science
Volume 1, Issue 7, July 2022
https://www.ijmdes.com | ISSN (Online): 2583-3138

Highly Scalable Recommendation System for Big Data Processing

Yoon So-Yun[1], Seong De[2], Sainath Chintareddy[3*]
[1]*Student, Department of Computer and Information Science, Pukyong National University, Pusan, South Korea*
[2]*Professor, Department of Computer and Information Science, Pukyong National University, Pusan, South Korea*
[3]*Professor, Department of Information Science, Vel Tech University, Chennai, India*

***Abstract*: With the development of networks and IT technologies, users are searching for and purchasing the items they want from anywhere, regardless of location. Accordingly, various studies are being conducted on how to solve the scalability problem caused by the rapidly increasing data in the recommendation system. In this paper, we propose an item-based collaborative filtering method to which tag weight is applied and a recommendation method using the MapReduce method, a distributed parallel processing method. The proposed method classifies items by category in the preprocessing process and groups them according to the number of nodes for speed and efficiency. Data processing is performed through 4 MapReduce steps in each distributed node, and item tag weights are used in the similarity calculation to recommend better items to users. The top N items among the predicted values output through the last Reduce step are used for recommendation. Through experiments, it was confirmed that the proposed method efficiently processes a large amount of data, and the suitability of recommendation is improved compared to the existing item-based method.**

***Keywords*: Recommendation technique, collaborative filtering, MapReduce, scalability, tags.**

1. Introduction

With the development of networks and IT technologies, the use of various mobile devices has become common. Therefore, users use mobile devices to find the items or information they want without any restrictions on time and place. Companies use a recommendation system to help users find valuable information about available services or products online according to the diversification of purchasing patterns. The recommendation system uses the received preference information to find the list of items that users prefer [1].

The collaborative filtering technique is the most used method [2] as a recommendation system. It extracts users with similar preferences to the target user and recommends items to the target user based on their preferences [3]. However, traditional collaborative filtering has a problem of scalability when analyzing or handling large-scale data [4], and since only the evaluation value of an item is used for prediction, it has a problem in that it cannot reflect the positive expression of buyers such as product reviews. [5].

The scalability problem, which is a disadvantage of collaborative filtering, is a problem connected to processing cost. To improve data processing cost and efficiency due to the recent explosive increase in data, Hadoop Map-Reduce, a big data distributed processing framework, was used. Various applied studies are being conducted [6]-[9]. In addition to the evaluation values of users, various studies are being conducted to analyze and utilize tag information such as product reviews, which are additional information left on products.

Recently, in [3] researchers developed a highly scalable MapReduce algorithm for collaborative systems that performs faster than traditional collaborative filtering algorithms. In this paper, we studied this scalable method and proposed more accurate recommendations to users. Tag information is applied to the collaborative filtering method, and to supplement the scalability problem, we propose a recommendation method that applies MapReduce, a distributed processing method.

The proposed method classifies items by category through a preprocessing process, calculates the similarity between items with a common evaluation number greater than or equal to a threshold, and designates the items with a similarity greater than or equal to the threshold as the nearest neighbor. Based on the evaluation values of the nearest neighbors, the predicted values for items not evaluated by the target user are calculated by applying tag weights, and the top N items with high predicted values are recommended to the user. The overall flow follows the recommendation step of the collaborative filtering technique, but in order to efficiently handle a large amount of data, the data is distributed among nodes, and then each node goes through 4 MapReduce steps to output the predicted value and recommend the item.

The proposed technique uses a distributed big data processing method to solve the scalability problem of collaborative filtering, while using a refined tag weight for each category item for more accurate item recommendation, improving the accuracy and efficiency of recommendation. it is possible

The structure of this paper is as follows. Section 2 describes the related studies, collaborative filtering and MapReduce, and section 3 describes the proposed technique. Section 4 evaluates the performance of the proposed technique using real data. Finally, Section 5 describes the conclusion.

*Corresponding author: generalgao2022@protonmail.com

2. Previous Works

A. Collaborative Filtering

The collaborative filtering algorithm is the most widely used recommendation system as a technique for recommending items or services to users by using the purchase information of other users who have similar preferences to the user's [2].

Collaborative filtering algorithms include user-based algorithms and item-based algorithms. The user-based algorithm designates users with similar preferences to the target user as nearby neighbors and recommends items based on their item evaluation values. The item-based algorithm measures the similarity between items and recommends an item based on the evaluation value of the item that is similar to the item preferred by the target all users.

$$sim(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \overline{r_i})(r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r_i})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \overline{r_j})^2}} \qquad (1)$$

$$P_{ut} = \frac{\sum_{i=1}^{c} R_{ui} \times Sim(t,i)}{\sum_{i=1}^{c} Sim(t,i)} \qquad (2)$$

Recommendation of collaborative filtering technique consists of 4 steps. In the first step, a user-item matrix is created based on the data that users evaluated, and based on this, the similarity between items is calculated in the second step. Based on the calculated similarity, the nearest neighbor is selected in the third step, and the predicted value is generated based on the evaluation value of the nearest neighbor in the last step, and the top N items are recommended to the target user. Equation (1) is a Pearson correlation coefficient equation that calculates the similarity between items. Eq. Equation (2) is a calculation formula that predicts the evaluation value of the unevaluated item t of the target users.

B. MapReduce

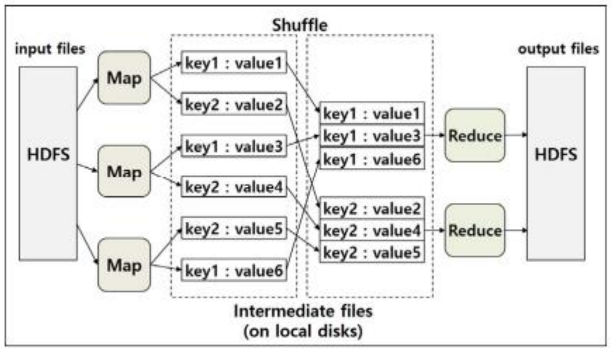


Fig. 1. MapReduce working

Hadoop is an Apache open-source project that provides a software framework for large-scale data processing and analysis. Both HDFS and MapReduce have a master/slave structure. In HDFS, the master is called a NameNode and the slave is called a DataNode, and can be used independently as a distributed file system.

MapReduce calls the master as a job tracker and the slave as a task tracker, and a distributed file system such as HDFS is absolutely necessary for reading and writing data. In addition, MapReduce consists of two stages: Map and Reduce. In the map stage, data in the form of <key, value> is received and processed to be converted into a new <key, value> and output.

In the reduce stage, a <key, list of values> that combines records with the same key value from the records output from the map is received and processed, and then a new <key, value> is output. Map and Reduce can be run on multiple machines, and the framework takes care of it [15]. Figure 1 shows the basic operation of Map and Reduce.

3. Recommendation Technique Applying Big Data Processing Technique

In this section, we describe the proposed method that applies the item tag weight and distributed processing method to the item-based collaborative filtering method to increase the accuracy and efficiency of recommendation. The proposed method consists of a pre-processing step, a prediction value calculation and a recommendation step, and in the second step, 4 maps and reduce are executed. Figure 2 is a schematic diagram of the system.

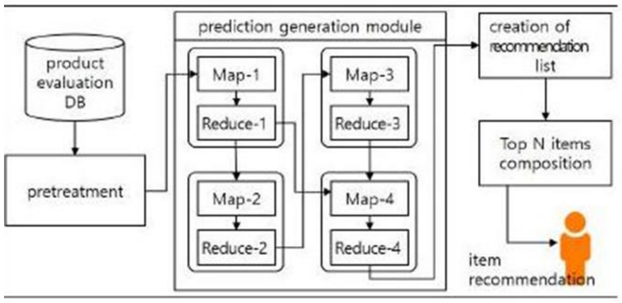


Fig. 2. System structure diagram

A. Preprocessing

Numerous items are registered in the recommendation system every day, and there are items that are continuously selected by users among the items, while there are items that are selected only for an initial period after registration and are not selected after that. In this paper, to improve the accuracy of recommendation, we classify items by category and then apply changes in users' evaluation values of the items over time. Determine the base date, calculate the value from the base date to the registration date, and if it is less than the threshold, use the formula to obtain the average of the existing item evaluation values. The average of the evaluation value is calculated using an expression that reflects both and the average of the evaluation value for a certain period, and only items with a result of 3 or higher are used.

In addition, the tag information on the item additionally expresses the user's preference for the item, and may affect other users' purchase of the item. However, users use very different expressions, and the same users may tag the same item multiple times or leave a tag for an item that has not been purchased. Therefore, in order to use the number of tags entered by users for an item as a weight in the calculation of similarity, the tags left on the same date with the same user ID for each item are designated once regardless of the number of tags, and the tags left without evaluation value are actually Excluded because purchase status is ambiguous. of calculating the average of the evaluation values to which time change is applied, denotes the number of users who evaluated the item during the entire period, and denotes the number of users who evaluated the item during the reference day.

$$\begin{cases} if\left(i_{fd} - i_{rd}\right) \leq \alpha, \\ \qquad \bar{R_i} = \dfrac{\sum_{j=1}^{l} R_{i,j}}{N} \\ elseif\left(\left(i_{fd} - i_{rd}\right) > \alpha \text{ and } N_p \geq \beta\right), \\ \qquad \bar{R_i} = \left(\dfrac{\sum_{j=1}^{l} R_{i,j}}{N} + \dfrac{\sum_{j=1}^{m} R_{i,j}}{N_p}\right)/2 \\ else\ 0 \end{cases} \tag{3}$$

The data is pre-processed under the above conditions and the (itemID, userID, rating, tagcount) values are saved as text files by category.

B. Prediction Value Calculation and Recommendation Using MapReduce

After the preprocessing step, the files stored for each category are subjected to 4 MapReduce steps to generate a predicted value.

The first MapReduce is the step of receiving preprocessed text and outputting the item ID, evaluation value, and tag count as a list for each user ID. The mapper sets the input file with userID as the key and (itemID, rating, tagCnt) as the value, and outputs the result. The reducer applies the inverse index of the tuples based on userID, and then lists (itemID, raing, tagCnt) based on userID to output. The following figure 3 shows the process of creating a list of items evaluated for each user ID.

```
procedure: Mapper-I
input : pdata(itemID,userID,rating,tagCnt) from m users to n items
output : tuples m𝔑 (<userIDs, (itemID,rating, tagCnt)..>)
  for each userID i in pdata
    key =userID
    value=(itemID,rating,tagCnt)
  end for
  emit tuples<key,value>

procedure: Reducer-I
input : tuples m𝔑 (<userIDs, (itemID,rating,tagCnt)..>)
output : tuples r𝔑 (<userIDs, Listitem>)
  Listitem ← List(itemID,rating,tagCnt)
  for each userID i=1 to m do
      templist ← new list
      for each userID j=1 to m do
        if i==j then
          add (itemID,rating,tagCnt) of tuples m𝔑 in templist
          Listitem=templist
      end for
  end for
  emit <userID, Listitem>
```

Fig. 3. Create list from user IDs

```
procedure: Mapper-II
input : tuples r𝔑 (<userIDs, Listitem>)
output : simCalList
  len = length(Listitem)
  simCalList ← a new list
  for each itemID i=1 to len do
      for each itemID j=i+1 to len do
        t=(itemID_i,itemID_j),(rating_i,rating_j,tagCnt_i,tagCnt_j)
        simCalList.push(t)
      end for
  end for
  emit simCalList

procedure: Reducer-II
input : simCalList
output : tuples s𝔑 (<(itemID_i,itemID_j),sim_ij>)
  for each itemID i=1 to len do
      compute the similarity of every to items using formula(4)
      key = (itemID_i,itemID_j)
      value = sim_ij
  end for
  emit tuples <key,value>
```

Fig. 4. Compute items similarity

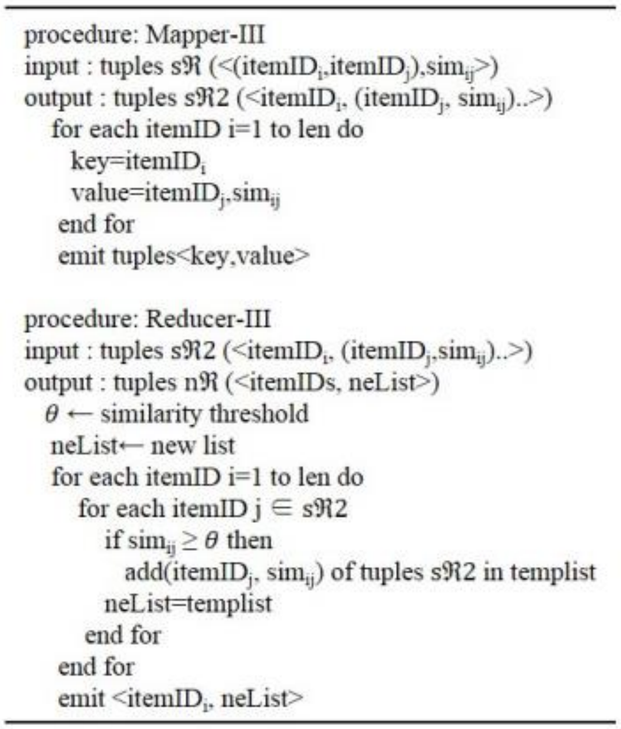
The second MapReduce is a multi-level system that calculates the similarity between two random items and outputs the result. Tuple <(itemID , itemID , rating using the evaluation value of all users who commonly evaluated any two items using the results of the previous multi-system mapping), tagcount rating. The reducer calculates the similarity between items to which the tag weight is applied using tuples, and then outputs <(itemID itemID tuples for any two items. Equation (4) is a similarity calculation formula. After finding the similarity using (1), the final similarity is calculated by applying the threshold value of the number of users for common evaluation and the tag weight, where is the number of users who commonly evaluated the items.

$$S(i,j) = \frac{\max|U(i \cap j), \gamma|}{\gamma} \cdot sim(i,j) \cdot w_t \tag{4}$$

The third MapReduce is the step of selecting the nearest neighbor item based on the output of the previous step. The mapper prints <itemID tuples for each item. The reducer sorts each item based on the similarity value, selects items with a similarity value greater than or equal to a threshold value as nearby neighbors, and outputs an (itemID) list for itemID. Figure 4 and Figure 5 show the process of calculating the similarity between items, respectively, and selecting the nearest neighbor based on the similarity.

Fig. 5 Selection of similarity-based neighborhoods. The fourth MapReduce is the step of calculating the predicted value using the item evaluation value and the similarity of the nearest neighbor to recommend the items not evaluated by the user and outputting a list. The mapper outputs a list of (itemID, rating, sim) based on userID using the result file of the first reducer and the result file of the third reducer. The reducer calculates the predicted value for the item that is not evaluated by uerID using Equation (2) and outputs <userID,list(itemID,) Figure 6

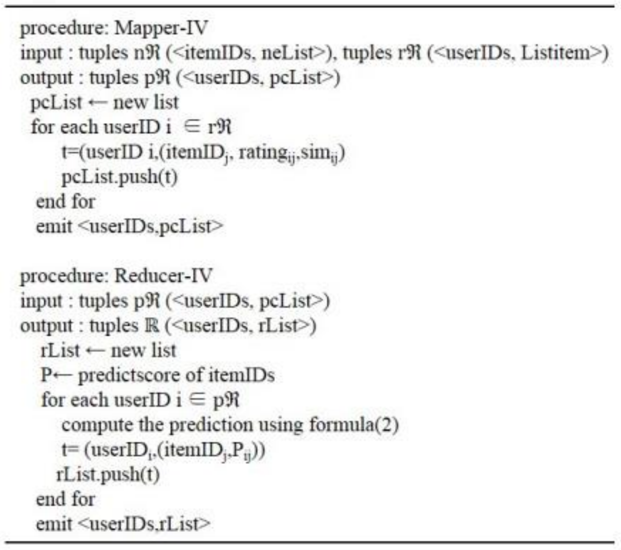
shows the process of calculating and outputting the predicted value. Finally, from the generated prediction value file, the item corresponding to the top dog with a high prediction value among the items not evaluated by the target user is extracted and recommended to the user.



```
procedure: Mapper-III
input : tuples s$\Re$ (<(itemID$_i$,itemID$_j$),sim$_{ij}$>)
output : tuples s$\Re$2 (<itemID$_i$, (itemID$_j$, sim$_{ij}$)..>)
    for each itemID i=1 to len do
      key=itemID$_i$
      value=itemID$_j$,sim$_{ij}$
    end for
    emit tuples<key,value>

procedure: Reducer-III
input : tuples s$\Re$2 (<itemID$_i$, (itemID$_j$,sim$_{ij}$)..>)
output : tuples n$\Re$ (<itemIDs, neList>)
   $\theta \leftarrow$ similarity threshold
   neList$\leftarrow$ new list
   for each itemID i=1 to len do
      for each itemID j $\in$ s$\Re$2
         if sim$_{ij} \geq \theta$ then
            add(itemID$_j$, sim$_{ij}$) of tuples s$\Re$2 in templist
         neList=templist
      end for
    end for
    emit <itemID$_i$, neList>
```

Fig. 5. Create a list of neighborhood



```
procedure: Mapper-IV
input : tuples n$\Re$ (<itemIDs, neList>), tuples r$\Re$ (<userIDs, Listitem>)
output : tuples p$\Re$ (<userIDs, pcList>)
  pcList $\leftarrow$ new list
  for each userID i $\in$ r$\Re$
       t=(userID i,(itemID$_j$, rating$_{ij}$,sim$_{ij}$))
       pcList.push(t)
   end for
   emit <userIDs,pcList>

procedure: Reducer-IV
input : tuples p$\Re$ (<userIDs, pcList>)
output : tuples $\mathbb{R}$ (<userIDs, rList>)
   rList $\leftarrow$ new list
   P$\leftarrow$ predictscore of itemIDs
   for each userID i $\in$ p$\Re$
       compute the prediction using formula(2)
       t= (userID$_i$,(itemID$_j$,P$_{ij}$))
     rList.push(t)
   end for
   emit <userIDs,rList>
```

Fig. 6. Calculate score and emit results

4. Experimental Results

The experiment is run in a Hadoop cluster environment consisting of one master node and five data nodes. Master node consists of Intel Core i5-5200u CPU 2.2GHz, memory 8GB, 250 Gbyte SSD, and data node consists of Intel Core i5-2400 3.1GHz, memory 4GB, 500 Gbyte disk. Server Java 1.8.0, Hadoop version, was installed on each node based on Ubuntu Linux 14.0. For the performance evaluation of the proposed method, the MovieLens data set collected by GroupLens was used. The data set has 95,580 tags and rating values of 10,681 movies by 71,567 users with a value of 15 preferences. In the experiment, a data set in a state in which the evaluation data was divided into training data and test data was used.

In the experiment, to evaluate the efficiency of the distributed processing method, speedup [16], which is the most used among various items to evaluate the scalability and efficiency of a parallel system, is used, and the F1-measure method is used to evaluate the suitability of the recommendation.

Speedup represents the change in efficiency according to the increase in the number of computational nodes. When the number of nodes is changed using the same data, it is effective when the result shows a linear relationship. Equation (5) is a speedup calculation formula, where T1 is the algorithm execution time in a single node, and Tp denotes the execution time in the nodes.

$$S_p = \frac{T_1}{T_p} \tag{5}$$

The technique is used to evaluate the recommendation performance and is calculated as a harmonic average of precision and recall. Precision is the ratio of items actually selected by the target customer among the list of recommended items generated by the recommendation system, and recall is the ratio of items recommended by the recommendation system among the items actually selected by the target customer. For precision and recall, when the number of recommendations increases, recall increases but precision decreases. It can be said that the higher the value, the better the suitability of the recommendation [17]. Equation (6) is an expression to calculate F1.

$$F_1 = \frac{2 \times recall \times preision}{recall + preision} \tag{6}$$

In this paper, to measure speedup performance according to item size, the number of items in the data set was configured differently to 10000, and the number of nodes was changed to 15 and an experiment was conducted.

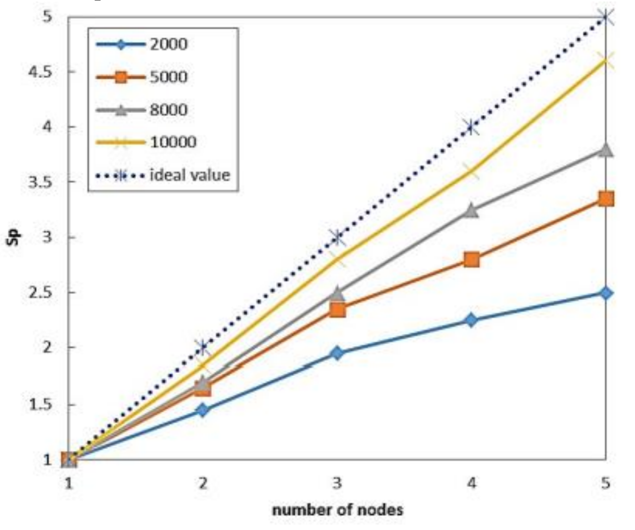


Fig. 7. Speedup result

Figure 7 shows the speedup experiment result and shows that speedup linearly increases as the number of items and nodes increases. In particular, it can be seen that the speed of adding nodes increases as the data set becomes larger than when the data set is small with the number of items being 2000, thereby improving scalability.

To measure the performance, a comparison experiment was conducted with the existing collaborative filtering techniques according to the change in the number of nearby neighbors. Figure 8 shows the experimental results according to the change in the number of nearby neighbors. It shows that the suitability of the recommendation of the proposed method is improved by about 6 22%.

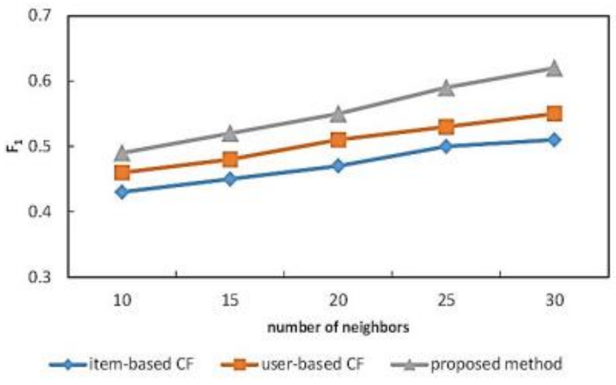


Fig. 8. F1 score

5. Conclusion

In this paper, in order to improve the scalability and recommendation suitability, which are problems of a collaborative filtering-based recommendation system, a big data distributed processing method and an item-based recommendation technique applied with tag weights were proposed. In the proposed method, data was preprocessed by applying changes in users' preference to items for accurate recommendations, and then the predicted values were output through MapReduce in four steps. In the similarity calculation, the method of designating the nearest neighbor through the calculation applying the tag weight was used. In an experiment to evaluate the scalability and suitability performance of the proposed method, the proposed method showed improved scalability when processing large-scale data and improved recommendation suitability compared to the existing collaborative filtering method. In future, we plan to study a method that is more suitable for real-time analysis and use tags left on items by users for analysis of items rather than just the number of tags, and to select similar items based on this.

References

[1] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of Netnews," Proceedings of the ACM Conference on Computer Supported Cooperative Work, pp. 175-186, New York, NY, USA, Oct. 1994.

[2] G. Linden, B. Smith and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," in IEEE Internet Computing, vol. 7, no. 1, pp. 76-80, Jan.-Feb. 2003.

[3] S. Manakkadu, S. Prasad Joshi, T. Halverson, and S. Dutta. "Top-k User-Based Collaborative Recommendation System Using MapReduce," in 2021 IEEE International Conference on Big Data (Big Data), pp. 4021-4025, 2021.

[4] A. Stanescu, S. Nagar, and D. Caragea, "A Hybrid Recommender System: User Profiling from Keywords and Ratings," 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies (IAT), pp. 73-80, Dec. 2013.

[5] Z. Zhao, M. Shang, "User based Collaborative-Filtering Recommendation Algorithms on Hadoop," 2010 Third International Conference on Knowledge Discovery and Data Mining, pp. 478-481, Jan. 2010.

[6] P. Ghuli, A. Ghosh, and R. Shettar, "A collaborative filtering recommendation engine in a distributed environment," 2014 International Conference on Contemporary Computing and Informatics (IC3I), pp. 568-574, Nov. 2014.

[7] Y. Shang, Z. Li, W. Qu, Y. Xu, Z. Song, and X. Zhou, "Scalable Collaborative Filtering Recommendation Algorithm with MapReduce," 2014 IEEE International Conference on Dependable, Autonomic and Secure Computing, pp. 103-108, Aug. 2014.

[8] F. Lu, L. Hong and L. Changfeng, "The improvement and implementation of distributed item-based collaborative filtering algorithm on Hadoop," 2015 34th Chinese Control Conference (CCC), 2015, pp. 9078-9083.

[9] H. Liang, Y. Xu, Y. Li, R. Nayak, and G. Shaw, "A Hybrid Recommender Systems based on Weighted Tags," in 10th SIAM International Conference on Data Mining (SDM 2010), Apr. 2011.

[10] S. Yun, S. Youn, "A Study on Recommender Technique Applying User Activity and Time Information," Journal of Korea Institute of Information and Communication Engineering, vol. 19, no. 3, pp. 543-551, Mar. 2015.

[11] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "GroupLens: Applying Collaborative Filtering to Usenet News," Communications of the ACM, vol. 40, no. 3, pp. 77-87, Mar. 1997.

[12] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based Collaborative Filtering Recommendation Algorithms," Proceeding of the 10th International World Wide Web Conference, ACM Press, pp. 285-295, May 2001.

[13] T. White, Hadoop: The Definitive Guide, Sebastopol. CA: O'reilly, K. Han, Do it: Hadoop with big data, Seoul: Easys Publishing.

[14] U. Ramachandran, H. Venkateswaran, A. Sivasubramaniam, and A. Singla, "Issues in understanding the scalability of parallel systems," in Proceedings of the First International Workshop on Parallel Processing, pp. 399-404. Dec. 1994.

[15] S. Ko, "A Recommender Agent using Association Item Trees," Journal of KIISE: Software and Applications, vol. 36, no. 4, pp. 298-305, Apr. 2009.