IJCCS (Indonesian Journal of Computing and Cybernetics Systems)
Vol.13, No.2, April 2019, pp. 149~158
ISSN (print): 1978-1520, ISSN (online): 2460-7258
DOI: 10.22146/ijccs.38596

Parallelization of Hybrid Content Based and Collaborative Filtering Method in Recommendation System With Apache Spark

Rakhmad Ikhsanudin*[1], Edi Winarko[2]
[1]Master Program of Computer Science; FMIPA UGM, Yogyakarta
[2]Department of Computer Science and Electronics, FMIPA UGM, Yogyakarta
e-mail: ***[1]rakhmad.ikhsanudin@gmail.com**, [2]ewinarko@gmail.com

Abstrak

Collaborative Filtering sebagai metode yang populer dalam sistem rekomendasi. Improvisasi dilakukan dengan tujuan untuk meningkatkan akurasi dari hasil rekomendasi. Salah satu cara yang ditempuh adalah dengan mengkobinasikannya dengan metode content based. Namun teknik penggabungan tersebut memiliki kekurangan dalam hal skalabilitas. Penelitian ini berusaha untuk mengatasi masalah skalabilitas yang dihadapi oleh sistem rekomendasi yang menggunakan metode hybrid collaborative filtering dan content based dengan menerapkan paralelisasi pada platform Apache Spark. Berdasarkan hasil pengujian, didapatkan nilai speedup metode hybrid collaborative filtering dan content based pada cluster Apache Spark dengan 2 node worker adalah sebesar 1,003 yang kemudian meningkat menjadi 2,913 pada cluster yang mempunyai 4 node worker. Nilai speedup meningkat kembali menjadi 5,85 pada cluster yang memuat 7 node worker.

Kata kunci— *sistem rekomendasi, kombinasi content based dan collaborative filtering, Apache Spark*

Abstract

Collaborative Filtering as a popular method that used for recommendation system. Improvisation is done in purpose of improving the accuracy of the recommendation. A way to do this is to combine with content based method. But the hybrid method has a lack in terms of scalability. The main aim of this research is to solve problem that faced by recommendation system with hybrid collaborative filtering and content based method by applying parallelization on the Apache Spark platform.Based on the test results, the value of hybrid collaborative filtering method and content based on Apache Spark cluster with 2 node worker is 1,003 which then increased to 2,913 on cluster having 4 node worker. The speedup got more increased to 5,85 on the cluster that containing 7 node worker.

Keywords— *recomendation system, hybrid content based and collaborative filtering method, Apache Spark*

Received Sept 6th,2018; Revised April 10th, 2019; Accepted April 28th, 2019

1. INTRODUCTION

In everyday life we are often faced with a large selection of items that we do not have knowledge of the item. In this case the recommendation system is present to provide recommendations on what items should be selected. The recommendations given are expected to help users determine what items will be chosen, such as what items to buy, what books will be read, what music will be heard or what films to watch and many more [1]. Collaborative Filtering is one of the popular algorithms used to build recommendation systems. Collaborative filtering generates recommendations based on the assessment or behavior of other users in the system. As a method that is widely adopted in the recommendation system, collaborative filtering is divided into 4 methods: user-based, item-based, model-based and fusion-based approach. In practice, collaborative filtering is also divided into three types, namely memory-based collaborative filtering, model-based and collaborative filtering. The working principle of a memory-based collaborative filtering algorithm is to use user ratings to get the same preference between users and between items [2].

Improvisation of collaborative filtering methods is done with the aim of increasing accuracy of the recommendations. One of the methods taken is to hybridize it with content based methods. Collaborative filtering generates recommendations based on active user ratings. Whereas content based methods improve recommendations based on items that have similarities to preferred items. This hybrid technique has proven to be superior to traditional recommendation techniques. However, despite having advantages in quality recommendations, the recommendation technique with the hybrid method has deficiencies in terms of scalability. Scalability is generally used in the technical domain to describe how system size and size of the problem will affect machine performance and algorithms. The number of data and algorithms are more complex, resulting in less optimal performance of the algorithm [3]. One indication of this scalability problem is the increased time needed to provide recommendations to users when the recommendation system data volume increases.

Scalability of a collaborative filtering recommendation system is a theme that is widely discussed in various studies with various proposed methods and approaches, one of which is the scale-out approach. In the scale-out approach, an additional computer node is used to run a recommendation system to obtain good scalability. The scale-out method implemented in previous research was using MapReduce Hadoop as practiced by [4], [5], [6], and [7] to get good scalability from traditional collaborative filtering recommendation systems. Another study was conducted by [8] who used Apache Spark to implement a scale-out approach to overcome the scalability of the recommendation system with traditional collaborative filtering methods. The use of Apache Spark by [8] was motivated by the assessment of the MapReduce Hadoop which used a lot of read and write processes to the hard disk which was considered less suitable for the implementation of collaborative filtering algorithms that have many iterative steps. It is expected that implementing it in the Apache Spark cluster will get more optimal results because Apache Spark is able to do processing using cache memory on each node in parallel.

2. METHODS

The architecture of the hybrid content based and collaborative filtering method models is shown in Figure 1. The first stage is reading a dataset consisting of three types of data, namely movies, ratings, and tags. Then the next step is to calculate the value of the bsed content method. After that, calculate the value of collaborative filtering methods. After the value of the calculation of the two methods is obtained, a hybrid calculation is performed using the results of the calculation of content based and collaborative filtering methods.

Furthermore, testing is carried out in parallel using the Apache Spark cluster. The model that has been created is run on each cluster scheme that has a number of different worker nodes. Then the acceleration obtained in each cluster scheme is calculated.

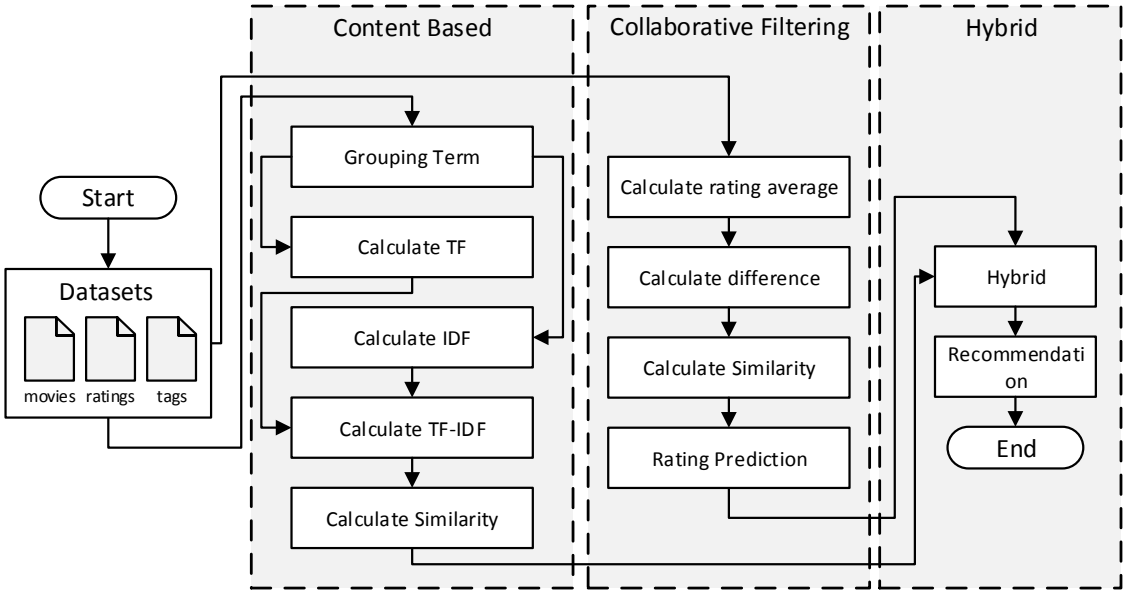

Figure 1 Hybrid collaborative filtering and content based architecture model

2.1 Content Based Filtering

Content based filtering method is usually used to look for similarities between documents using the terms contained in the item. But in this study, content based filtering will be used to calculate the similarity of a movie using genres and tags as terms. Then the preference is divided by combining genres and tags from the movie that the user likes. Then it will be compared with each movie that has no rating. The similarity of items to preferences greatly affects the value obtained by the item.

1. The first step of the content based method is to classify terms. Performed by calculating the number of terms that appear on each movieId.
2. Then the second step is to calculate the TF (Term Frequency) phrase in each movieId using equation (1). $f(d,t)$ is occurrence of the word t in document d and $n(d)$ is the number of words or terms contained in the document d.

$$TF(d,t) = \frac{f(d,t)}{n(d)} \quad (1)$$

3. After that, calculate IDF (Inverse Document Frequency) from each phrase. Performed using equation (2). $df(t)$ adalah jumlah dokumen yang memiliki kata t dan D adalah jumlah seluruh dokumen. IDF mempertimbangkan frekuensi kata pada seluruh dokumen yang ada. Pembobotan IDF menganggap bahwa bobot sebuah kata akan besar jika kata tersebut sering muncul dalam sebuah dokumen tetapi tidak banyak dokumen yang mengandung kata tersebut.

$$IDF(t) = log\left(\frac{D}{df(t)}\right) \quad (2)$$

4. Furthermore, calculating TF-IDF is done by calculating the TF value of the phrase in each movie multiplied by the IDF phrase value. Shown in equation (3). $TF(d,t)$ is the term frequency of a word or term in a document and $IDF(t)$ is the inverse document frequency of the term term.

$$TF\ IDF(d,t) = TF(d,t) * IDF(t) \quad (3)$$

5. Calculate similarity using the cosine similarity approach as shown in equation (4). The cosine similarity approach is often used to determine the proximity between text documents [9].Cosine similarity is a calculation that measures cosine values from the angle between two vectors (or two documents in a vector space). The results of the dot product addition to the TF-IDF phrase value for each movieId with the TF-IDF term values on the preference. Then divided by the value of the square roots of the sum of the results of the squared TF-IDF term in the movieId, multiplied by the sum of the results of the squared TF-IDF term in the preferences.

$$Similiarity(i_x,\ i_y) = \frac{\sum_{j=1}^{U} x_j . y_j}{\sqrt{\sum_{j=1}^{U} x_j^2\ \sum_{j=1}^{U} y_j^2}} \quad (4)$$

2.2 Collaborative FIltering

Collaborative filtering method uses rating as the basis for rating prediction. In this study using collaborative filtering with an item based approach. Item-based collaborative filtering algorithm was developed to cover the weaknesses found in user-based collaborative filtering [10]. The basic idea is to make items that have been rated by users as a basis for calculating similarity, then a group of items that have similarity are selected with items that have been rated by the user. The similarity value is used as a weight when predicting the rating value on the target item. Users will get a movie recommendation that has a tendency similar to other users.

1. The first step is to calculate the average rating of each movie as shown in equation (5). The amount of rating (x_i) in the movie is divided by the COUNT rating value in the movie (n).

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (5)$$

2. Then the difference between the average and the mean values is calculated. Shown in equation (6).

$$difference = x_i - \bar{x} \quad (6)$$

3. To calculate the Pearson-correlation value between two items, all rating values that do not have a partner with the same user are removed from the account. For example the set of users who give a rating on two items i and j is U, then the pearson-correlation equation to calculate the similarity of items i and j or $s(i,j)$ is shown in equation (7). $R(u,i)$ is the rating value given by the user to item i, while $R(u,j)$ is the rating value given by the user to the item j. $\bar{R}(*,i)$ is the average rating given in item i and $\bar{R}(*,j)$ is the average rating given in item j. U is a set of users who have given a rating on items i and items j.

$$s(i,j) = \frac{\sum_{u\in U}\left(R(u,i) - \bar{R}(*,i)\right)\left(R(u,j) - \bar{R}(*,j)\right)}{\sqrt{\sum_{u\in U}\left(R(u,i) - \bar{R}(*,i)\right)^2}\ \sqrt{\sum_{u\in U}\left(R(u,j) - \bar{R}(*,j)\right)^2}} \quad (7)$$

4. At the prediction stage, [10] proposes a weighted sum algorithm to predict as shown in the equation (below). As the name implies, the calculation of predictions for rating on item i by

user u, written $P(u,i)$, is done by adding up all rating values that are item-neighborhood members. Each sum added is weighted with $s(i,j)$, which is the similarity value of item i with item j. As shown in equation (8). $s(i,j)$is the similarity value between item i and item j. $R(u,j)$ is the rating given by the user to item j. N is an item-neighborhood set.

$$P(u,i) = \frac{\sum_{j \in N} s\,(i,j)\; R(u,j)}{\sum_{v \in N}(|s(i,j)|)} \tag{8}$$

5. The last stage is recommendation, which is sorting based on predictive values and then selecting a number of items that have the highest predictive value. These recommended items have never been rated by active users, so after getting these recommendations the user is asked to provide feedback in the form of rating values.

2.3 Hybrid

This hybrid technique combines the results of calculating several linear recommendation techniques. This merger calculates the rating prediction separately first, then the results of each method are combined into one. [11] uses the weighted average formula to combine these results. In this study will apply a linear combination of methods. However, the combination that will be used is by summarizing the product of each method and its weight. Shown in equation (9). R_{hybrid} is prediction value. w_n is weight of the method that used. R_n is value of method calculation.

$$R_{hybrid} = (w_1 R_1 + w_2 R_2) \tag{9}$$

2.4 Testing

In this study, clusters will be created using Google Cloud Dataproc services. Then the data will be stored on the Google Cloud Storage service. The architecture of the cluster created is shown in Figure 2.

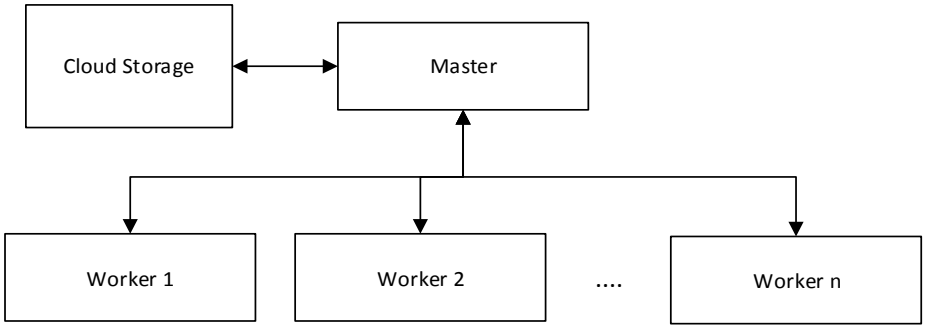

Figure 2 Apache Spark cluster architecture

The series of stages were executed in series starting from the first stage to the last stage. One stage can stand alone or process results from the previous stage. The dataset in the cluster is divided into several partitions, computing on stage is done on each partition in the form of a task. The task is run in parallel by the executor on each node. Drivers communicate with a coordinator called a master, who manages workers to run executors. Worker or slave is an instance that contains the executor to run the task. After SparkContext is connected to the cluster manager, the executor is allocated to each node to run the process and store data. Then the program code is sent to the executor, and finally SparkContext sends the task to the executor to run. As shown in Figure 3.

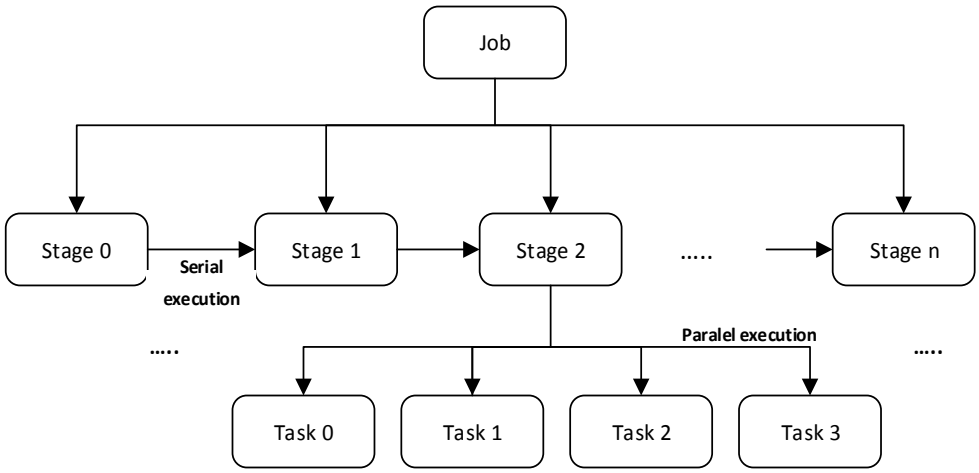

Figure 3 Job paths in the Apache Spark cluster [12]

3. RESULTS AND DISCUSSION

3.1 Dataset

The data used for testing is the opensorce dataset obtained from MovieLens. The dataset used to generate recommendations consists of 671 users, 9066 movies, 1,222 tags and 100,004 record ratings. Dataset movie consists of three columns, namely movieId, title, and genres. The movieId column contains the movie id, the title column contains the movie title, and the genres contain movie streams separated by "|". Dataset ratings consists of four columns, namely userId, movieId, and rating. UsertId contains a user id that gives a rating. Then the movieId contains a movie id that is rated. While the rating contains the value of the movie rating given by a user. Dataset Tags consists of three columns, namely userId, movieId, and tags. UserId contains user IDs that tag a movie. Then the movieId contains the movie id tagged by the user. While the tag contains the tag phrase that the user gives to a movie.

3.2 Experimental Environment

The machine specifications used are n1-standard-1 which has one virtual CPU (2.3 GHz Intel Xeon E5), 100 GB storage, and 3.75 GB RAM. For testing purposes, several cluster schemes are used, each of which has one master node, but the number of different worker nodes is 2, 4, and 7 node workers. Other configurations that have not been mentioned use the default configuration. After creating the create process, it takes some time for the cluster creation process. Only need to wait until the cluster status is ready.

For comparison, a computer with a single node is used with the specifications of one virtual CPU (2.3 GHz Intel Xeon E5), 500 GB storage, and 3.75 GB RAM.

3.3 Calculation results of Hybrid Method

Of all the calculations, ten movieId were taken which had the biggest final score. The final value is obtained from the value of the collaborative filtering calculation multiplied by its weight then added to the content based value multiplied by its weight. As explained in equation (9). In addition to the results of collaborative filtering calculations, the division with a value of 5 is obtained to get a value range scale similar to the content-based method, namely 0-1. The final result of the calculation will have a value range of 0-2. The greater the final result, the more recommended.

The calculation results for each experiment that is done do not always find the same value. Often there are different rounding values. However, the difference does not affect the recommendation results due to movieId with the top ten order not changing.

Table 1 Calculation Results hybrid collaborative filtering and content based methods

No	MovieId	Content Based (weight=1)	Collaborative Filtering (divided by 5)(weight=1)	Hybrid
1	**70336**	0,66717449109678	0,74491634673289	**1,41209083782967**
2	**2054**	0,61050189855768	0,80000000000000	**1,41050189855768**
3	**58025**	0,68081597297213	0,71534288023423	**1,39615885320636**
4	**32825**	0,58441558619341	0,80000000000000	**1,38441558619340**
5	**117529**	0,68081597297213	0,69965882007545	**1,38047479304758**
6	**41569**	0,64893071080044	0,71798528066884	**1,36691599146928**
7	**136016**	0,56406788846900	0,80000000000000	**1,36406788846899**
8	**6537**	0,61292775385261	0,75051285216518	**1,36344060601778**
9	**85179**	0,59518070272305	0,76342020534217	**1,35860090806522**
10	**61248**	0,66717449109678	0,69108733680345	**1,35826182790022**

3.4 Experimental results on the cluster

To find out the scalability of hybrid content based and collaborative filtering methods that are run in clusters, speedup is calculated using equation (10). S_p is speedup of cluster. T_a is running time average of cluster with smallest number of worker. T_p is running time average of cluster that speedup will be calculated.

$$S_p = \frac{T_a}{T_p} \tag{10}$$

There several cluster schemes that have the number of workers for testing purposes, namely 1, 2, 4, and 7 workers. These data are processed in each cluster scheme using hybrid collaborative filtering and content based methods. Testing is carried out ten times in each cluster scheme. So that the total of all experiments conducted was 40 times as shown in Table 2. The "w" column represents the number of workers and the column "TRY (second)" represents each experiment performed.

Table 2 Runtime testing hybrid collaborative filtering and content based methods

W	*TRY* (*second*)									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	607	3158	2080	3923	2617	6359	11076	20425	252201	27298
2	543	3779	2466	3555	3177	6464	10654	20180	25584	27441
4	238	1176	793	1458	1205	3065	4642	5291	8132	9703
7	270	687	640	667	676	1357	2032	2227	3721	5522

On each data and cluster size the average value is calculated. Then calculate the speedup value on each data size by using the division operation between its average value with the average value of the cluster that has the smallest number of workers, in this case the cluster with the number of workers one as shown in equation (10).

Appear in Table 3 the increase in speedup is obtained as the number of node workers increases. Speedup is obtained between Apache Spark's standalone runtime on the Apache Spark cluster with 2 workers that are relatively the same, namely 1.003, cluster speedup with 4 workers of 2.913, and speedup back in the cluster with 7 workers of 5.85.

Table 3 The results of the calculation of the speedup method of hybrid collaborative filtering and content based

Worker	Average Execution Time	Speedup
1	10412,2	1
2	10384,3	1,003
4	3574,8	2,913
7	1779,9	5,85

Examples of speedup calculations in clusters with 7 workers are shown in equations (11) and (12). S_7 is a speedup obtained by a cluster with 7 workers. T_1 is the average cluster execution time with 1 worker. Then T_7 is the average cluster execution time with 7 workers. Figure 4 shows the speedup graph of the combination method of collaborative filtering and content based on the apache spark cluster.

$$S_7 = \frac{T_1}{T_7} \tag{11}$$

$$S_7 = \frac{10412,2}{1779,9} = 5,85 \tag{12}$$

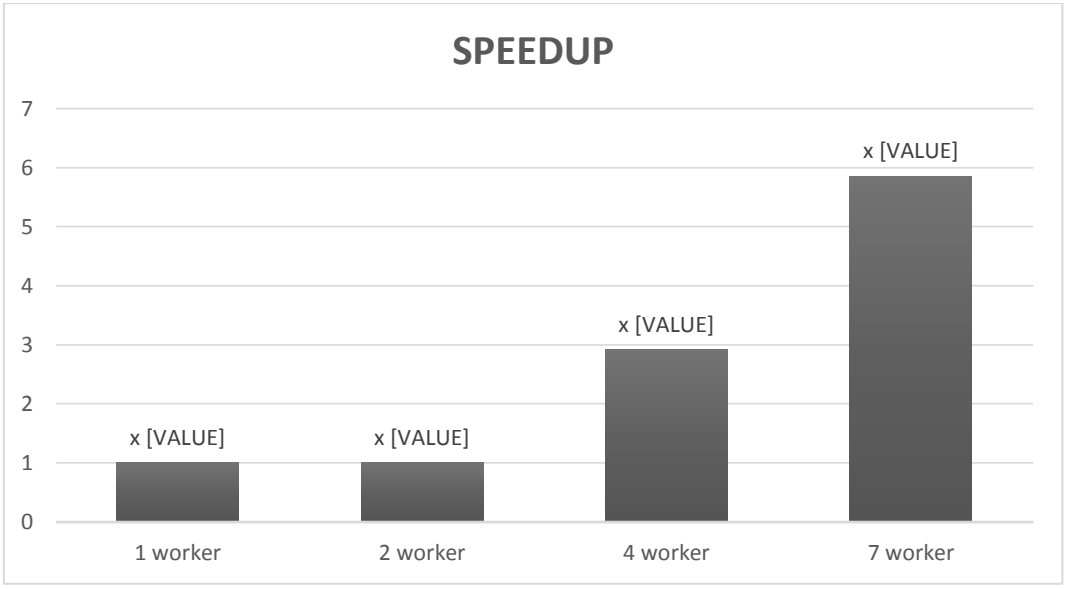

Figure 4 Speedup graph

4. CONCLUSION

The parallelization using Apache Spark on hybrid collaborative filtering and content based methods gets good runtime results. Shown in increasing speedup obtained in each cluster scheme as the number of workers increases. Obtained the speedup on the cluster scheme with 2 workers that is equal to 1.003 with an average runtime of 10384.3 seconds, speedup on the cluster scheme with 4 workers of 2.913 with a runtime of 3574.8 seconds, and an increase in speedup again found in the cluster scheme with 7 workers of 5 , 85 with a runtime of 1779.9 seconds.

REFERENCES

[1] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*. 2015.

[2] A. Segal, Z. Katzir, and K. Gal, "EduRank : A Collaborative Filtering Approach to Personalization in E-learning," *Proc. 7th Int. Conf. Educ. Data Min.*, no. Edm, pp. 68–75, 2014.

[3] P. Wang, H., Zhang, "Hybrid Recommendation Model Based on Incremental Collaborative Filtering and Content- based Algorithms," *2017 IEEE 21st Int. Conf. Comput. Support. Coop. Work Des.*, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8066717/. [Accessed: 2- Sept- 2018]

[4] S. Schelter, C. Boden, and V. Markl, "Scalable similarity-based neighborhood methods with MapReduce," in *Proceedings of the 6th ACM conference on Recommender systems - RecSys '12*, 2012, p. 163.

[5] P. Ghuli, A. Ghosh, and R. Shettar, "A Collaborative Filtering Recommendation Engine in a Distributed Environment," *Proc. - 2014 Int. Conf. Contemp. Comput. Informatics*, pp. 568–574, 2014.

[6] Y. Shang, Z. Li, W. Qu, Y. Xu, Z. Song, and X. Zhou, "Scalable Collaborative Filtering Recommendation Algorithm with MapReduce," in *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, 2014, pp. 103–108.

[7] P. A. Riyaz and S. M. Varghese, "A Scalable Product Recommendations Using Collaborative Filtering in Hadoop for Bigdata," *Procedia Technol.*, 2016.

[8] E. Casey, "Scalable Collaborative Filtering Recommendation Algorithms on Apache Spark," Claremont McKenna College, 2014.

[9] W. G. S. Parwita, "Hybrid Recommendation System Memanfaatkan Penggalian Frequent Itemset dan Perbandingan Keyword," vol. 9, no. 2, pp. 19–21, 2015.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the tenth international conference on World Wide Web - WWW '01*, 2001, pp. 285–295.

[11] M. Claypool, T. Miranda, A. Gokhale, P. Murnikov, D. Netes, and M. Sartin, "Combining content-based and collaborative filters in an online newspaper," *Proc. Recomm. Syst. Work. ACM SIGIR*, pp. 40–48, 1999.

[12] N. Bharill, A. Tiwari, and A. Malviya, "Fuzzy Based Clustering Algorithms to Handle Big Data with Implementation on Apache Spark," in *Proceedings - 2016 IEEE 2nd International Conference on Big Data Computing Service and Applications, BigDataService 2016*, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7474361/. [Accessed: 2- Sept- 2018]