I.J. Intelligent Systems and Applications, **2018, 7, 74-81**
Published Online July 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2018.07.08

An Improved Hybrid Distributed Collaborative Filtering Model for Recommender Engine using Apache Spark

Rakesh K. Lenka
Department of Computer Science & Engineering, IIIT Bhubaneswar, India
E-mail: rakeshkumar@iiit-bh.ac.in

Rabindra K. Barik
School of Computer Application, KIIT University, India
E-mail: rabindra.mnnit@gmail.com

Sasmita Panigrahi and Sai S. Panda
Department of Computer Science & Engineering, IIIT Bhubaneswar, India
E-mail: sasmita239@gmail.com, saipsiddhant@gmail.com

Received: 23 June 2017; Accepted: 15 September 2017; Published: 08 July 2018

Abstract**—**The present scenario there is a serious need of scalability for efficient analytics of big data. In order to achieve this, technology like MapReduce, Pig and HIVE came into action but when the question comes to scalability; Apache Spark maintains a great position far ahead. In this research paper, it has designed and developed an improved hybrid distributed collaborative model for filtering recommender engine. Execution time, scalability and robustness of the engine are the three evaluation parameters; has been considered for this present study. The present work keeps an eye on recommender system built with help of Apache Spark. Apart from this, it has been proposed and implemented the bisecting KMeans clustering algorithms. It has discussed about the comparative analysis between KMeans and Bisecting KMeans clustering algorithms on Apache Spark environment.

Index Terms**—**Apache Spark, Recommendation Engine, Collaborative Filtering, Machine learning, KMeans, Bisecting KMeans, Bigdata.

I. Introduction

In this dynamic era, the technology is ever updating. A small add-on counts a lot in the next accomplishment of work/project. There are up to 'n' parameters that are taken care of, but even after that as per the field requirements necessary changes are brought up and released as new distributions, projects and in some cases it's contributed to the existing model. For consideration of big data analytics, the recommendation system has been growing tremendous popularity in the field of e-commercial marketplace. It has been greater benefit for the businesses grow by taking accurate business strategies and policies by suitable analysis through recommendation engine. Initially for recommendation engine, Collaborative Filtering (CF) has used as one of the efficient technique which depends on the past user transaction and feedback data [1-3].

Latent factor models and neighborhood approaches have been used in the recommendation engine for the traditional CF algorithms. There are several issues like Scalability, Sparsely and Cold start problems have been found in the traditional CF. For solving these issues, there are different kind of approaches have been implemented. Feature based recommendation engine using tagging, Clustering and hybrid techniques are the various technical approaches being used for the solving out the problems associated on the recommendation engine. But these approaches are not suitable for massive big dataset. Recently, various works have been done with parallisation CF algorithm in Hadoop Environment. But it has been found that the cost efficient and greater computation time in MapReduce of Hadoop framework. By keeping this in mind, Apache Spark has been used for a new hybrid solution for recommendation engine with traditional CF methods by combing both dimension reductionality and KMeans clustering methods of machine learning [4-5].

In the present research paper, it has developed the Bisecting KMeans clustering methods of machine learning by taking the same dataset which has been previously implemented [9]. Comparison analysis between KMeans and Bisecting KMeans has been illustrated. Thus, the present study has been categorized into the number of sections. The section 2 deals with the related works which has been done in the field of recommendation engine. Section 3 illustrates about the objective of the present research and the detail description about the data set and proposed model representation. Section 4 describes the parallel implementation of Bisecting KMeans clustering on Apache Spark. Section 5 contains the comparison analysis and observations of KMeans and Bisecting KMeans clustering. A final conclusion is drawn in Sections 6 with future avenues of research.

II. Related Works

Recommender Systems have developed a lot since their evolution. From the period of its origin Recommender System are based upon three forms i.e. demographic, content based and collaborative filtering, but currently they are even utilizing social information retrieved from public domain like Facebook, Twitter, etc. This paper gives an outline of recommender systems as well as collaborative filtering methods and algorithms and investigates about its evolution, offers an authentic classification, provides area of future works and aware about development in specific areas selected from past, present or future importance [10].

Recommender System have already created an important place at everyone's life may be implicitly or explicitly. The job of a Recommender System is to automatically find and suggest items to a person of his/her interest, at times it even identifies his/her needs. Here, it also discusses the task of recommender system and the typical methods that never utilize the social network info. The discussion leads to how social network information accepted by recommender system as supplementary input for attaining better precision. Here, It has been found that the CF-based social recommender systems. These social based recommended systems have been categories into matrix factorization based social recommendation approaches and neighborhood based social recommendation approaches. For each and every observance, there are several representative algorithms have been taken into consideration [11]. Collaborative Filtering has become the most preferred solution while designing a recommender system. With the drastic enhancement of storage and network technology, the quantity of users and items in recommendation system has been raised; here an effectual method of user based collaborative filtering on MapReduce has been shown. This paper works on Bag of Word (BoW) method and designs a hierarchical inverted index to upsurge the scalability. Interim, a soft assignment tool for the hierarchical inverted index has been discussed resulting in fixation of decrease in accuracy of recommendation caused by the index. The MapReduce are implemented in both real data and simulated data, proving the implementation has the potential to scale to vast number of items and users which has been assured the recommendation accuracy [7, 23].

Despite knowing the computational complexity of Collaborative Filtering is high and inefficiency in large scale system, the present study uses it broadly in many fields. Over here, it has been taken this issue into consideration and performs a user based Collaborative Filtering algorithm on Cloud Computing environment, specifically Apache Hadoop, for solving the reliability and scalability problem of Collaborative Filtering. From the experimental observations, it infers a technique that divides users into clusters as per two essential principles that is orderly arrangement of mapper number to overcome the initiation of mapper and divide the task uniformly such that all processors has to finish task at the identical time, that leads to achieve speedup the processes [5].

One of the very important steps in developing the recommender system for any purpose is to select the foundational framework. There are plethoras of approaches to it, the framework can be established from the ground up, or an existing recommender system can be utilize the several code libraries which has been adapted, or a framework may be selected and tailored to suit (Apache Mahout, MymediaLite, LensKit, etc.). However the best approach has to use an existing free and open source environment which have active contributors, it has numerous libraries and if required functions can be added and built [12, 24].

Recommendation system is a system used for predicting the right preference to the user. Over here it has been scrutinized an improved hybrid recommendation algorithm and combine MapReduce program on Hadoop environment. With the usage of this hypothesis the improved algorithm can precisely attaining user preferences, offer the required recommendation during the period when the user surfs the web page. In the end, Hadoop meets the massive data processing and attain extremely high performance system by use of the data reprocessing method [8].

Recommendation provides a feature of automatically identifying the taste of the user based on the prior patterns, interest shown by him/her. In this paper, it proposes a recommendation system to manage huge data available at web in form feedback, of complaints, remarks, ratings, reviews, opinions and comments about any item i.e. event, product, services and individual by using Apache Hadoop Framework. It has been executed interface of Mahout for examining the data supplied by the review and the rating site for movies [13].

With the enormous amount of facts present on web, the analysis of finding the relevant news to a user is a tire some job, which cannot accomplished by manually picking the News. Therefore in the present study, it has been justified the need of recommender system tailor made for recommending news articles. The developed recommender system continuously analyzes a data-stream using Apache Flink framework, and provides real-time recommendations. The recommender system understands the type of news portals and the categories of the audience of it and updates the recommender model assuring that only new articles are recommended. The architecture of the systems is discussed followed by the hurdles of processing continuous streams. The scalability and the methods for optimizing the parameter configuration are explained [14].

Apache Spark is well known for it’s in memory computation and has achieved a suitable place in many designs, hypothesis and models. Today many of the companies, labs and various units are keenly interested to port their existing technology to Apache Spark [26]. With the rapid development of the big data in recent years, several applications have been extended for processing in big data analytics framework in health and geospatial area [15, 16, 25]. Apache Spark is an open source distributed computing framework which has been based on MapReduce of Hadoop algorithms. It has been offering a good platform to integrate MapReduce, Machine Learning, SQL, Streaming and Graph Processing. Apache Spark is much faster than that of MapReduce for the executing programs due to the use of Resilient Distributed Datasets (RDDs) which are evenly distributed collection of objects. These blocks split into numerous partitions. These partitions have been again computed in parallel on different nodes of the cluster. The new Data Frames of API has been introduced in Spark 1.6.0. It has been even performing faster than that of RDDs and has been provided SQL operations on RDDs. Spark 1.6.0 has two types of programs i.e. Driver Program which run on Master and Worker Program which runs on Slave. Spark Context is responsible for the connection to the Cluster Manager which in turn allocates the resources on the slave(worker) nodes which has been shown in Fig. 1 [17].

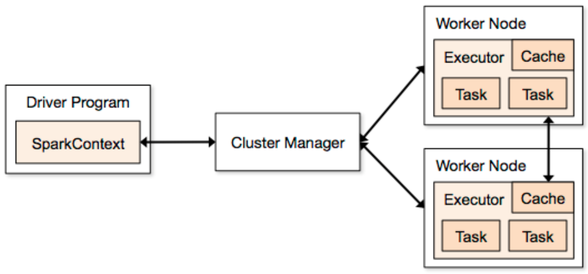

Fig.1. Cluster node overview [26]

Broadcast variables and accumulators are the two varieties of shared variables have been used in Spark. Broadcast variables have used to sore values in memory over all the nodes where as in accumulator; the variables can be added to counters and sums. Due to the Lazy evaluation process of Spark, the efficiency has been improved that means actions have been appraised and the alterations have been kept for upcoming implementation of tasks. A new RDD has been constructed due to transformations process by some previous conditions and actions have computed the results based on the RDD which has been returned to the driver program or has been saved to the external storage system. Collaborative Filtering recommender systems utilize information regarding user's predilection to provide bespoke predictions. In this paper, it has described about an algorithmic framework built over Apache Spark for parallel computation of the neighborhood based Collaborative Filtering problem, where algorithm linearly satisfied the cumulative number of users. It has been worked on plenty of variation on this methodology including correlation and vector-based similarity calculations, user and item based recommendation approaches, and selective down-sampling of user interactions. In the end, it has taken an extensive research on comparison of these methods on the Movie Lens dataset consisting of 10 million movie ratings [18, 19].

Recommending the news article has always been a matter of concern as the user always finds a different categories of interest in News, may be supporting the trending news at one point of time or putting an immense interest on business or economy page. Traditional Recommender approaches are optimized for analyzing static data set. In news recommending strategies continuous changes, high volume of messages, and tight time constraints, alternative strategies are needed. Therefore, it has designed a highly scalable recommender system optimized for the processing of streams. It has been observed that the model in the CLEF NEWS-REEl challenge. The model is built on Apache Spark enabling the distributed processing of recommendation requests ensuring the scalability of the proposed approach [21].

Recommender systems have become an area of active research. They are especially important in the ecommerce industry because they help increase revenues and improve customer experience. A lot of different techniques have been explored for different kind of recommender systems based on the desired objective and the data that is available to base the recommendations on. In the present work, it has been proposed and developed a simple recommender system based on hybrid solution to the user based Collaborative Filtering based on Apache Spark platform combining both dimensional reduction and clustering techniques. This work also emphasizes on fixing of issues like cold start problem by relating users to products via features obtained. Here the tags are used as features [22].

III. Aim of the Current Research and Proposed Model Representation

The aim of the current research paper is to offer and implement the bisecting KMeans clustering algorithms on Apache Spark Environment with Movie Lens dataset in an existing hybrid distributed collaborative model for filtering recommender engine. It has also compared with the existing KMeans clustering algorithm and illustrated the different observation with respect to execution time, scalability and robustness of the engine.

The present research, it has been taken the data from Movie Lens [20]. It takes 20M dataset that it processes 20 million ratings rated by users. For scalability aspects, it has been used 1M & 10M dataset including 1 million and 10 million ratings. A user has the right to rate and tag a movie rating can be done on a scale of range 1 to 5 considering 1 as lowest and 5 as highest. Collaborative filtering is used to identify different users which are similar to a particular user. Then these similar users are used to recommend products that were popular with them. In the problem, it has been worked on; the notion of users is very weak because, it has no additional information about the users except for the ratings they have rated.

Also the number of tags per user is very low.

Hence, it has decided to use collaborative filtering replacing the notion of user by search queries and try to identify similar queries to each query in the training set. It has also used the notion of finding similar items to what was searched for using the new query and recommend popular items among similar items. Firstly data loaded to hive and pertinent features are taken out. User ratings with less than 30 counts in rating movies are driven out, movies possessing less than 3 are deleted and all tags are lower cased and stop words are detached. The fresh dataset framed is represented in Table 1.

Table 1. Pre-processing dataset [20]

List of Attributes	Users	Tags	Movies
Before	138493	465000	24744
After	110615	441252	16409

After obtaining the pre-processed data, the model has been built and stored to HIVE5 in parquet format. All these computation jobs have done in offline environment. In real scenario, it has been loaded the models rear from Hive which has again used to create top N recommendations. Hence, it has been increased the throughput of the developed recommender system. The existing model is divided into two parts namely prevailing user module and fresh user module. The block diagram of the existing user recommendation module has been defined in Fig. 2.

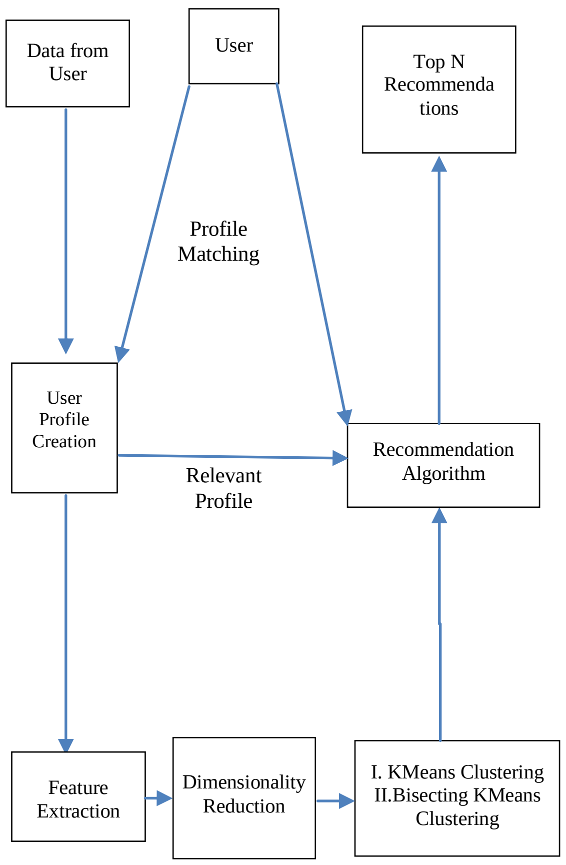

Fig.2. Block Diagram: Existing-User Recommender Module

In the new user recommender model which has been illustrated in Figure 3, it has been found that the cluster of the user which has belong to the inputs given by the user. Subsequently, it has been suggested that the top N highly rated movies of that cluster has not seen by relevant users; has been returning as the recommendation. Users can easily list down the tags which they are concerned in and accordingly the proposed model; it finds the ideal recommendation for the users for avoiding coldstart problem.

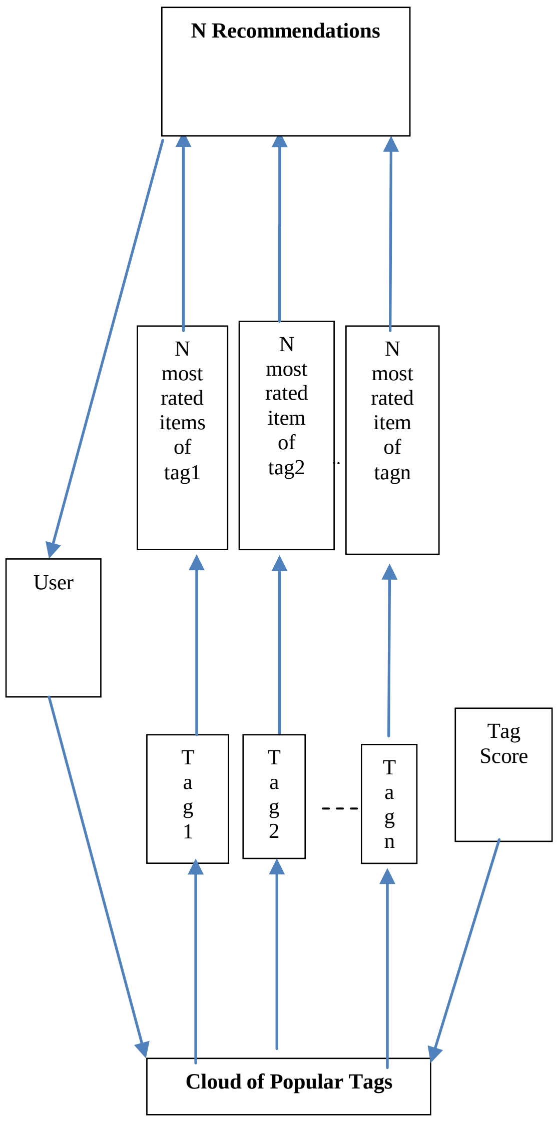
N Recommendations
N most rated items of tag1
N most rated item of tag2
N most rated item of tagn
User
Tag1
Tag2
Tagn
Tag Score
Cloud of Popular Tags

Fig.3. Block Diagram: New-User Recommender Module

With the usage of K Means clustering algorithm similar users are grouped as per set built by Alternating Least Square (ALS) model. In ALS model, it has been implemented the bisecting K Means till, it has obtained a minimal squared error function which has been derived from equation 1.

$$F=\sum_{i=1}^{k}\sum_{j=1}^{n} \|x_j^{(i)} - c_i\|^2 \quad (1)$$

In Equation 1, "||xj – ci|| " is defined as the Euclidean distance between xj and ci, 'xj' is equivalent to the number of data points in jth cluster, where 'c' has been shown the number of cluster centers. Where users most have closest to their respective cluster center. The desired users are those who act as representatives of the cluster. These users have been defined as the most effective and desired relevant users on the top of the sysetm. Initially, it has been retrieved the top k most relevant and effective users of each cluster and has been stored it into the desired Hive table.

The fresh new users have assimilated the labels easily and labels can be served as bridge gap helping desired users for better distinguish an unidentifiedassociation between an element and themselves. Initially, Tag Score (TS) for each tag has been calculated. TS(t,m) has been defined for a Tag (t) and Movie (m) in equation 2.

$$TS(t,m) = \frac{Number\ of\ times\ t\ has\ been\ applied\ to\ I}{\sum Number\ of\ times\ any\ tag\ applied\ to\ i} \quad (2)$$

The fresh new user has the responsiblility to pick the labels from which they has been identified the list and as per the preference, the most pertinent top N items which has related to the favourite tags have been returning as recommendation to the identified user.

IV. Implementation of Bisecting Kmeans Clustering

Here, it has been described the execution of the projected effort on Apache Spark. The projected procedures have been written in Scala programming language. Initially, it has been introduced the two files as rating and tag files. These rating file(rating_file.csv) and tag file(tag_file.csv) are stored into the HDFS file system of Hadoop Framework. The execution of Apache Spark has been started by generating a sparkContext object. As data would be frequently accessed, it has been cached it in memory. Initially, it has been proposed the dimensiton reduction algorithm which has been illustrated in Algorithm 1.

Algorithm 1 Algorithm for Dimensionality Reduction

```
Input: Rating File (rating_file.csv) [User_ID, Movie_ID, User_Rating]

Output : User_Feature<User_ID, Feature_Vector>
         Product_Feature<Movie_ID, Feature_Vector>
Begin :
         On each of the worker node is doing in parallel process:
  1.  Load_Data: the data from rating_file.csv file into an RDD.
          data← load(rating_file.csv)
  2.  Parse_Rating: It is known as user defined function which has been splitting the desired data based upon comma (',') and has been returned as an RDD of Rating class object.
          Parse_Rating←map(Parse_Rating)
          Emit_Data<Rating(User_ID, Movie_ID, User_Rating)>
  3.  Store_Data: the Parse_Rating data into the memory by using cache() function
  4.  Random_Split() the RDD into training_RDD and test_RDD where 80% split in training_RDD and 20% split in test_RDD.
  5.  Do_map on test_RDD and store the first two fields into another RDD.
          test←map(User_ID, Movie_ID, User_Rating)
  6.  Emit_Data <User_ID, Movie_ID>
  7.  While a= 1 to i            [i presents the number of iterations]
  8.  While b = Array (1 to j)    [which is containing different values of 'λ']
  9.  While c = Array (1 to k)    [which is containing different values of Rank]
  10. model←ALS.train(train_RDD)
  11. predict←model.predict(test)
  12. Error←map(calculate_RMSE)
  13. Emit_Data<Error>
              end while
              end while
              end while
  14. While the values of b and c, which has given the Least Error (RMSE), do again step 10.
  15. Emit_Data<User_Feature, Product_Feature>
  16. Store the results in Hive tables
      model.saveAsParaquetFile("ALSmodel.Paraquet")
```

The output of the above ALS algorithm has been passed as the input to next KMeans clustering and Bisecting KMeans clustering algorithm. The following Algorithm 2 and Algorithm 3 have been derived the existing K Means clustering and proposed Bisecting Kmeans algorithms.

Algorithm 2 Algorithm for KMeans Clustering

```
Input  : User_Feature<User_Id, Feature_Vector>
Output : Top N Recommendation
Begin  :
  1. Master broadcasts the User_feature to all of the worker nodes.
     On each of the worker node has doing in parallel process: Normalise the Feature_Vector for all users.
  2. Normalise←FeatureVector.ComputeColumnSummaryStatistics()
  3. Emit_Data <mean, variance>
  4. While a = 1 to i, i = Number of iterations
  5. While b = 1 to j,   j = Number of clusters
  6. cluster = kmeans.train(User_Feature_Vector)
```

```
                    end while
                    end whle
7.  While each User_Feature:
8.  Cluster_ID←model.predict(User_Feature)
9.  Cluster_Center←model.clusterCenters(cluster_ID)
10. distance←computeDistance(User_Feature,cluster_Center)
11. Join the Movie_IDs has been keyed on User_IDs from
        Parse_Rating data RDD. [Step 3 of ALS]
12. Emit_Data<(Cluster_ID),Array(User_ID,Movie_ID)>.take_Ordered(N)
13. While any active user AU
    [AU→<(Cluster_ID),(User_ID,Movie_ID)>]
14. AU←map(Top N Recommendations)
        Where top N is the user defined function
      which will return the top N recommendations
    14.1 filter() the common Movie_IDs between
         AU and relevant
            user set emitted from step 11.
    14.2 Emit_Data <Array(Movie_ID)>.top N
         where Movie_ID are the
            top N highest rated movie by relevance
                    end while
```

Algorithm 3 Algorithm for Bisecting KMeans Clustering

```
Input: C: Number of clusters presents
       D: Top N documents obtained by vector space similarity
Output: C clusters put all the N documents in a single cluster
Begin: put all the N documents in a single cluster K
1.  While a=1 to C-1 do for b=1 to ITER do
2.  While b=1 to C-1 do for b=1 to ITER do
3.  Use KMeans to split K into two sub clusters, K1 and K2
4.  if ( intra cluster similarity (K1) > intra cluster similarity(K2) )
5.  make cluster K1 as permanent
6.  K= K2
7.  else
8.  make cluster K2 as permanent
9.  K = K1
10. end if
11. end while
12. end while
13. end Bisecting KMeans
```

Intially, User_Feature vector has been disseminated to each slave node. The user feature_vector normalized with Compute_Column_Summary_Statistics function. It has been computed in column wise summary statistics. It has been used MLLib of Spark 1.6.0 both KMeans and Biseting KMeans algorithm to train the proposed model. The compute_Distance() has been computed the distance of user_Feature vector to its cluster_Center. After KMeans and Bisection KMeans Clustering computation, it finally computes the Tag Score (TS). In new user, the user chooses the labels at once, most pertinent objects are returned as reference based on the TS, which has been described in Algorithm 4.

Algorithm 4 Algorithm for Computing Tag Score.

```
Input : tag_file.csv [User_ID, Movie_ID, Tag]
Output : <User_ID, Movie_ID, Tag, Tag_Score>
Begin:
1. Each worker node has been done in parallel process:
   Repeat step 1 to 3 of ALS algorithm as input.
2. Data_Frame←ParseTag.DF()
3. Data_Frame.registerTempTable("Tag")
4. Val_ordered_ID = sqlContext.sql("SELECT Movie_ID AS ID, TAG FROM TAG
   ORDER BY Movie_ID")
5. Val_eachTag_Count = ordered_ID.groupBy("ID,TAG").count()
6. Val_final_result = sqlContext.sql("SELECT Movie_ID, TAG_NAME, occurrence AS
   eachTag_Count, count AS total_Count FROM result ORDER BY Movie_ID")
7. Val_Tag_Score = sqlContext.sql("SELECT Movie_ID,
   TAG_NAME(eachTag_Count/total_Count) AS TagScore FROM final_result"))
```

V. Comparison Analysis and Observations

The observations have been taken on operating system-Ubuntu 14.04 environment which has specifications such as 2.50 GHz processors with 4 processing cores. 4GB RAM has been allocated to master node of clusterwhere4GB RAM allocated to each slave node. The recent technologies such as Apache Hadoop 2.7.2, Scala 2.11.7, Apache Hive 2.0, Apache Spark 1.6.0 and SBT 0.13.9 has been used for implementation. The mentioned model has been executed and tested for various parameters, among the considered parameters the most effective parameter observed as the different execution time across various datasets 1M, 10M, and 20M. Table 2 shows the execution time with KMeans and Bisecting KMeans Algorithm with various datasets with 6 numbers of clusters.

Table 2. Comparison of execution time

Data Set	KMeans	Bisecting KMeans
1M	40ms	38ms
10M	45ms	40ms
20M	50ms	44ms

Figure 4 shows the bar chart of the above execution times as compared to KMeans and Bisecting KMeans.

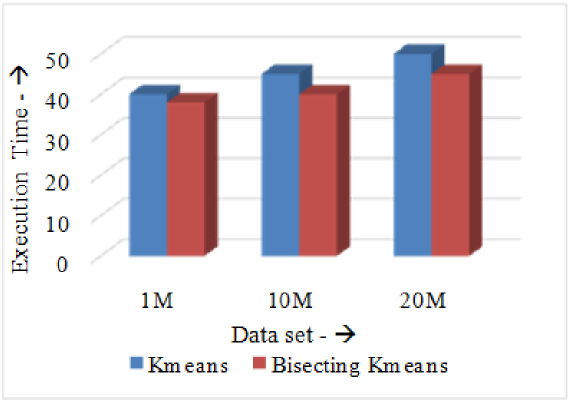

Fig.4. Comparative Analysis of KMeans and Bisecting KMeans Clustering Algorithm with various dataset and execution times

From the above analysis it has been concluded that Bisecting KMeans is a better approach for the problem being discussed. It has also observed that there is a significant difference of executing the model comes into action if the data is more and more voluminous. Therefore, it will be appropriate if it can use the discussed model for real world data analytics.

VI. Conclusion and Future Work

The discussed and projected model has been verified with several datasets of 1 million, 10 million & 20 million; also found as more effectual than prevailing models. The experimental result shows that the running time of algorithm is boosted with every new node addition into the Spark Cluster. Our model is generating best output as compared to standard algorithms in terms of throughput. Further we also have an inclusion of widespread review of merits and demerits of all Collaborative Filtering algorithms and found that our model performs the superlative among all. Few challenges identified can be higher needs of RAM size by Spark memory computation which is costly. In order to speed computational time, we select Scala (Spark's native language). Scala is initially perplexing but their functional programming feature makes it substantial. To get enhanced estimate, we update the ALS model and Clusters in manual mode. Subsequently, it has been strategize to devise this model expending Mahout (A Machine Learning Library of Hadoop) to perceive the scalability and efficiency of model.

References

[1] Shapira, Bracha, Francesco Ricci, Paul B. Kantor, and LiorRokach. "Recommender Systems Handbook", 2011.

[2] Ricci, Francesco, LiorRokach, and Bracha Shapira. Introduction to recommender systems handbook. Springer US, 2011.

[3] Casey, Walker Evan. "Scalable Collaborative Filtering Recommendation Algorithms on Apache Spark", 2014.

[4] Hashem, Ibrahim Abaker Targio, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. "The rise of "big data" on cloud computing: Review and open research issues." Information Systems, Vol. 47 pp. 98-115, 2015.

[5] Zhao, Zhi-Dan, and Ming-Sheng Shang. "User-based collaborative-filtering recommendation algorithms on hadoop." In Knowledge Discovery and Data Mining, 2010.WKDD'10. *Third International Conference on*, pp. 478-481. IEEE, 2010.

[6] Mo, Yijun, Jianwen Chen, Xia Xie, Changqing Luo, and Laurence Tianruo Yang. "Cloud-based mobile multimedia recommendation system with user behavior information." *IEEE Systems Journal* 8, no. 1, pp. 184-193, 2014.

[7] Shang, Yang, Zhiyang Li, WenyuQu, YujieXu, Zining Song, and Xuefei Zhou. "Scalable collaborative filtering recommendation algorithm with MapReduce." In Dependable, Autonomic and Secure Computing (DASC), *2014 IEEE 12th International Conference on*, pp. 103-108. IEEE, 2014.

[8] Wang, Chunzhi, Zhou Zheng, and Zhuang Yang. "The research of recommendation system based on Hadoop cloud platform." In Computer Science & Education (ICCSE), *2014 9th International Conference on*, pp. 193-196. IEEE, 2014.

[9] Panigrahi, Sasmita, Rakesh Ku Lenka, and Ananya Stitipragyan. "A Hybrid Distributed Collaborative Filtering Recommender Engine Using Apache Spark." *Procedia Computer Science*, Vol. 83, pp. 1000-1006, 2016.

[10] Bobadilla, Jesús, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. "Recommender systems survey." *Knowledge-Based Systems, Vol.* 46, pp.109-132, 2013.

[11] Yang, Xiwang, Yang Guo, Yong Liu, and Harald Steck. "A survey of collaborative filtering based social recommender systems." *Computer Communications*, Vol. 41, pp. 1-10, 2014.

[12] Walunj, Sachin Gulabrao, and Kishor Sadafale. "An online recommendation system for e-commerce based on apache mahout framework." In Proceedings of the 2013 annual conference on Computers and people research, pp. 153-158. ACM, 2013.

[13] Verma, Jai Prakash, Bankim Patel, and Atul Patel. "Big data analysis: recommendation system with Hadoop framework." In Computational Intelligence & Communication Technology (CICT), 2015 *IEEE International Conference on*, pp. 92-97. IEEE, 2015.

[14] Ciobanu, Alexandru, and Andreas Lommatzsch. "Development of a News Recommender System based on Apache Flink." In Working Notes of the 7th *International Conference of the CLEF Initiative*, Evora, Portugal. 2016.

[15] Barik, Rabindra K., Harishchandra Dubey, Arun B. Samaddar, Rajan D. Gupta, and Prakash K. Ray. "FogGIS: Fog Computing for Geospatial Big Data Analytics." arXiv preprint arXiv:1701.02601 (2016).

[16] H. Dubey, J. Yang, N. Constant, A. M. Amiri, Q. Yang, andK. Mankodiya, “Fog data: enhancing telehealth big data through fog computing,” in Proceedings of the ASE BigData & SocialInformatics2015. ACM, , pp. 14, 2015.

[17] Han, Zhijie, and Yujie Zhang. "Spark: A Big Data Processing Platform Based on Memory Computing." In Parallel Architectures, Algorithms and Programming (PAAP), 2015 Seventh *International Symposium on*, pp. 172-176. IEEE, 2015.

[18] Ding, Dongliang, Dongyue Wu, and Fuli Yu. "An overview on cloud computing platform spark for Human Genome mining." In Mechatronics and Automation (ICMA), 2016 *IEEE International Conference on*, pp. 2605-2610. IEEE, 2016.

[19] Maarala, AlttiIlari, Mika Rautiainen, MiikkaSalmi, Susanna Pirttikangas, and JukkaRiekki. "Low latency analytics for streaming traffic data with Apache Spark." In Big Data (Big Data), 2015 *IEEE International Conference on*, pp. 2855-2858. IEEE, 2015.

[20] Harper, F. Maxwell, and Joseph A. Konstan. "The movielens datasets: History and context." *ACM Transactions on Interactive Intelligent Systems (TiiS)*, Vol. 5, no. 4, pp.19, 2016.

[21] Domann, Jaschar, Jens Meiners, Lea Helmers, and Andreas Lommatzsch. "Real-Time News Recommendations Using Apache Spark." In Working Notes of the *7th International Conference of the CLEF Initiative*, Evora, Portugal. 2016.

[22] Zhao, Zhi-Dan, and Ming-Sheng Shang. "User-based collaborative-filtering recommendation algorithms on hadoop." In Knowledge Discovery and Data Mining, 2010.WKDD'10. *Third International Conference on*, pp. 478-481. IEEE, 2010.

[23] Wang, Chunzhi, Zhou Zheng, and Zhuang Yang. "The research of recommendation system based on Hadoop cloud platform." In Computer Science & Education (ICCSE), 2014 9th *International Conference on*, pp. 193-196. IEEE, 2014.

[24] Ciobanu, Alexandru, and Andreas Lommatzsch. "Development of a News Recommender System based on Apache Flink." In Working Notes of the 7th *International Conference of the CLEF Initiative*, Evora, Portugal. 2016.

[25] Gupta, N., Lenka, R. K., Barik, R. K., & Dubey, H. "FAIR: A Hadoop-based Hybrid Model for Faculty Information Retrieval System." *arXiv preprint arXiv:1706.08018* (2017).

[26] Lenka, R. K., Barik, R. K., Gupta, N., Ali, S. M., Rath, A., & Dubey, H. "Comparative analysis of SpatialHadoop and GeoSpark for geospatial big data analytics." Contemporary Computing and Informatics (IC3I), *2016 2nd International Conference on*. IEEE, 2016.

Authors’ Profiles

Rakesh K. Lenka Rakesh K. Lenka has completed his M.Tech in Computer Science & Engineering from Motilal Nehru National Institute of Technology, Allahabad. Currently he is working as an Assistant Professor in the department of Computer Science and Engineering of IIIT Bhubaneswar and pursuing Ph.D. in VSSUT, Burla. His research areas of interest include Machine Learning, Computational Intelligence, Data Mining and Cloud Computing. He is a member of IEEE and CSI.

Dr. Rabindra K Barik is currently working as an Assistant Professor in the School of Computer Applications, KIIT University, Bhubaneswar, India. He has received his both M.Tech and Ph.D. in Geoinformatics from Motilal Nehru National Institute of Technology, Allahabad, India. His research area includes Geospatial Database, SOA, Cloud Computing, IPR and Geoinformatics. He is a member of IEEE and IAENG.

Sasmita Panigrahi has completed his M.Tech in Computer Science & Engineering from IIIT Bhubaneswar. Her research areas of interest include Machine Learning, Big data. Computational Intelligence and Data Mining.

Sai Siddhant panda is an enthusiast in the field of Machine Learning and is an undergraduate from IIIT Bhubaneswar. He holds a Bachelors of Technology Degree in computer science engineering. He completed his degree in the year 2017. He successfully completed a couple of months internship at a media channel start-up as a blog developer in a company “OdishaLive”, situated at Bhubaneswar, currently he is working as a Research Intern at CloudThat Technologies PVT LTD. He can be reached by saipsiddhant@gmail.com.

How to cite this paper: Rakesh K. Lenka, Rabindra K. Barik, Sasmita Panigrahi, Sai S. Panda, "An Improved Hybrid Distributed Collaborative Filtering Model for Recommender Engine using Apache Spark", International Journal of Intelligent Systems and Applications(IJISA), Vol.10, No.7, pp.74-81, 2018. DOI: 10.5815/ijisa.2018.07.08