Collaborative Filtering based Movie Recommendation System using Apache Spark

Raghad Rowaida
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
raghadrowaida@cmail.carleton.ca

December 14, 2019

Abstract

This paper studies different modern techniques for building a high performing movie recommendation systems. Study showed that Collaborative Filtering (CF) algorithm worked most efficiently for movie recommendation[1] . A special variety of CF algorithm named Alternating Least Square (ALS) method is applied for the development of the system. ALS algorithm runs in multi-core environment which is implemented in Apache Spark framework.

1 Introduction

Computation speed of a process was limited to the clock speed of the processor. Parallel computing changed this notion. The idea of parallel computing was surfacing in the form of supercomputers since late 60s. But it entered a whole new level when IBM introduced world's first multi-core computer by in 2001. Computation speed is no longer limited to the clock speed of a single core processor. Multiple cores are present to divided workload and work parallel to enhance computational performance. From general perspective it seems pretty straight forward. But in reality parallel computing does not enhance performance linearly with the growth of cores. There are several issues that influence performance in a parallel environment. Dynamic load balancing between multiple active cores is a big challenge. Memory latency, bandwidth, etc decrease performance. Also there are several tasks which are fundamentally impossible to parallelize. That is why parallel computing is a big research area. New and improved techniques are proposed to enhance the computational performance in parallel environment for different types of problem sets.

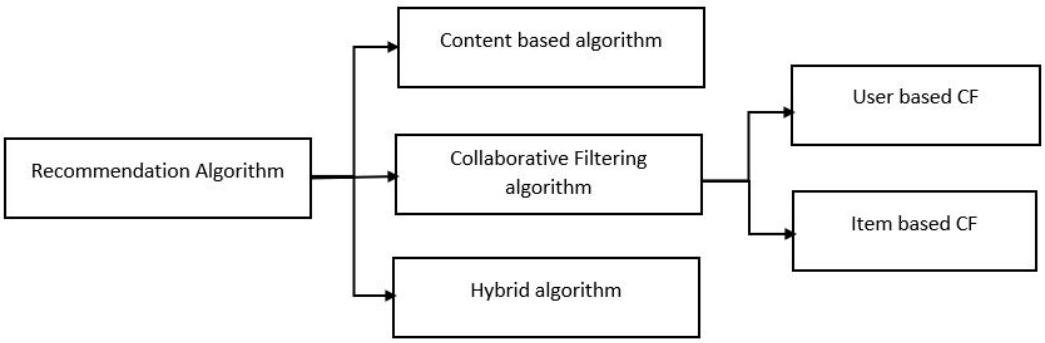
The focus of this study is to find a high performing algorithm to construct a recommendation system for movies[2]. Recommendation system is an important business tool for today's fast growing data driven world. All online service based companies like Youtube, Netflix, Spotify are sustaining because of their huge user based. These vast variety of users are continuously interacting with these systems to get their desired items. As the system grows, the number of users and the number of items grows too. A very efficient way to keep a user engaged to the system is to guide the user to an item that s/he will appreciate. That is why all these big companies invest a lot of research work to develop an appropriate

recommendation system. But building a well performing recommendation is challenging. Big systems have huge collection of items and users. So recommendation system needs to be capable of dealing with a huge data collection. But the recommendation algorithm is expected to work in real time. Unless the recommendation system works in real time, it will not be able to keep a user continuously engaged with the system. Sequential calculation is not suitable for processing this huge collection of user-item data. Parallel computing is essential to develop a fast working recommendation system. This project developed a movie recommendation system that works in a multi-core parallel environment and able to recommend a suitable movie to it's user in real time. A special class of recommendation algorithm named Collaborative Filtering (CF) is used to develop this project. The recommendation application runs on Apache Spark framework. Project used MovieLens ml-100K dataset. It achieved satisfactory performance.

The paper is organized in several sections. Section 2 will review the relevant literature for this work. Section3 explains the problem statement of the project. Proposed solution is discussed in the next section, section 4. Section 5 and 6 cover respectively, experimental evaluation, and conclusion.

2 Literature Review

According to [2][3], algorithmic approaches to build a recommendation system can be classified in to below categories:



- **Content based algorithm** focuses on user profile information and object profile information. Then try to match them accordingly.
- **Collaborative filtering (CF)** [4] methods considers past history of users' preferences, activities, and behaviors and recommend items based on the similarities to other users. It is so far the most efficient system of them all.
- **Hybrid algorithm** is a composition of Content-based and CF method.

Collaborative Fitting itself is a vast area of recommendation system. Two sub-classes of CF algorithm are (a) User based CF, and (b) Item based CF.

a **User based CF** method start from user side and cluster them based on their similar interests in products. Then from the recorded user rating to similar products the

system recommend the non-mapped items to the users in the same cluster. Several approaches are used to carryout the CF algorithm. Principle Component Analysis (PCA), Matrix Factorization are some approaches for CF[5]. Matrix Factorization technique [6]is a machine learning approach which has multiple implementation. A very efficient variety of Matrix Factorization is Singular value decomposition (SVD)[7]. This method was used in Netflix prize 2006 and won the competition. But this method works sequentially. A better implementation of Matrix Factorization is Alternating Least Square (ALS) algorithm. This one runs in parallel fashion. That is why is works faster than regular SVD approach.

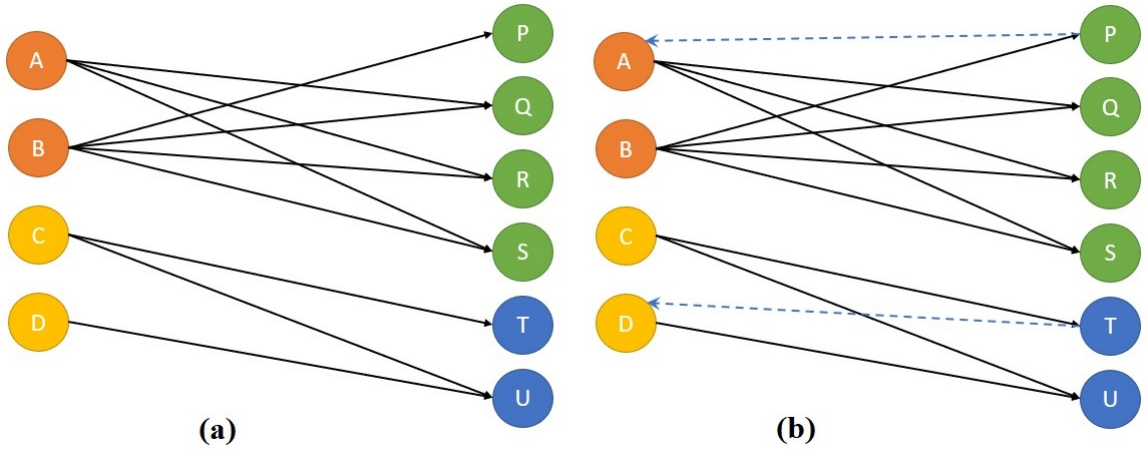


Figure 1: (a) Clustering the users based on similar interests on items. (b) Recommendation based on user similarity

- **Matrix Factorization** technique takes the user to item rating data as input represented in a 2D matrix form. Then decomposes the input matrix into two matrices as shown in figure 2. Matrix Factorization is a machine learning technique that tries to minimize a fitness function[7]. The purpose of the fitness function is to minimize the difference between recorded rating and expected rating of a user-item relation in the input matrix. To perform the matrix factorization a fixed number of latent factor is needed. This latent factor [8] can be different types of related features. For a movie recommendation system LF1 can be user rating, LF2 can be movie genre, LF3 can be release year, etc. Choosing the number of latent factor is a very crucial job. If the number of latent factor is very small, then the system will suffer from under fit problem. Example: if user based CF Matrix Factorization is implemented with one latent factor, then it will actually work as a Item based CF. On the other hand if the number of latent factor is too many, then it will suffer from over fitting problem. After choosing a suitable latent factor number Matrix Factorization uses the fitness function and applies optimization technique like, gradient descent algorithm to find the cell values of the decomposed matrix. SVD approach works very well to perform Matrix Factorization in this way. The only short coming is that SVD cannot work in parallel. That is why it takes longer time to compute. Other very effective way to perform Matrix Factorization is ALS approach.

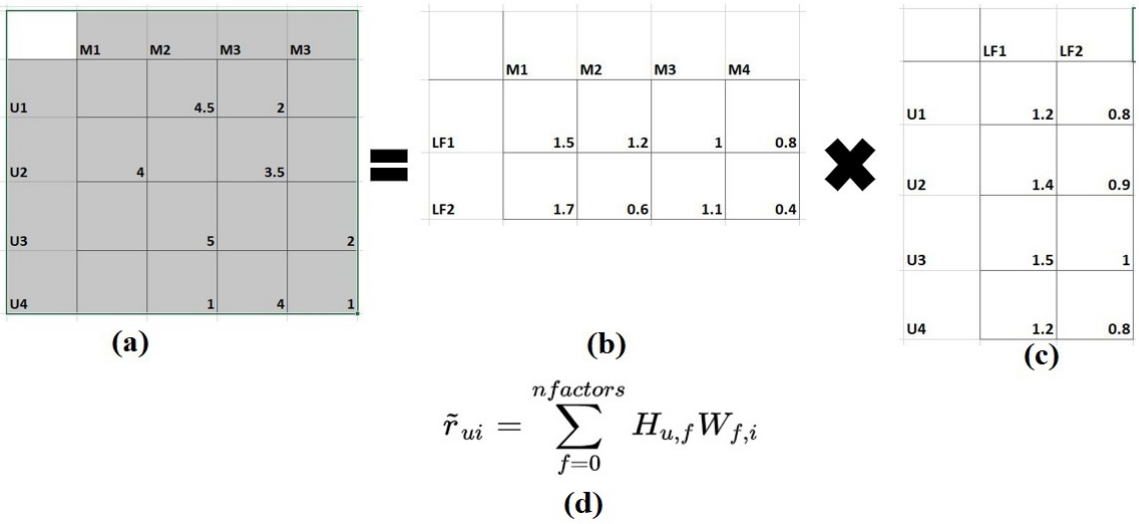


Figure 2: Matrix Factorization (a) Input matrix, (b) Decomposed left matrix where rows indicates the number of latent factors (here two latent factors LF1 and LF2 is used), columns indicates the items, and each cell represents the optimized ratings relation between item and latent factors, (c) Decomposed right matrix where rows indicates the users, and columns represents latent factors, and each cell represents the optimized ratings relation between user and latent factors (d)Equation representation of matrix Factorization

$$\arg\min_{H,W} \|R - \tilde{R}\|_{\mathrm{F}} + \alpha\|H\| + \beta\|W\|$$

Figure 3: Example fitness function that needs to be minimized

- **Alternating Least Square (ALS)** method is a special parallel approach to perform matrix factorization where instead of minimizing a single fitness equation globally it uses two different perspective minimization. ALS minimizes two loss functions alternatively; It first holds user matrix fixed and runs gradient descent with item matrix; then it holds item matrix fixed and runs gradient descent with user matrix.ALS runs its gradient descent in parallel across multiple partitions of the underlying training data from a cluster of machines.

b **Item based CF** [8]usually uses clustering algorithms like K-Nearest Neighboring (KNN) to cluster similar rated items from the items' side and recommend them to the users. It is a good recommendation algorithm with some shortcomings. Item based CF suffers from cold-start problem. Also it suffers from popularity bias[9] [10].

Now the involvement of parallel computing in the above procedures can be in many forms. Mostly the Matrix Factorization process is divided into individual operations which are computed in parallel. Both CPU and GPU computing is possible for the calculation. Different big data mining frameworks like Hadoop can be used in case of Content-based analysis, Data pre-processing in CF, or in Hybrid approaches. The knowledge-based recommendation algorithms are working for the 'surprise' factor, that provides rich recommends outside of users' expectation. Machine learning is also playing a role in recommendation systems.

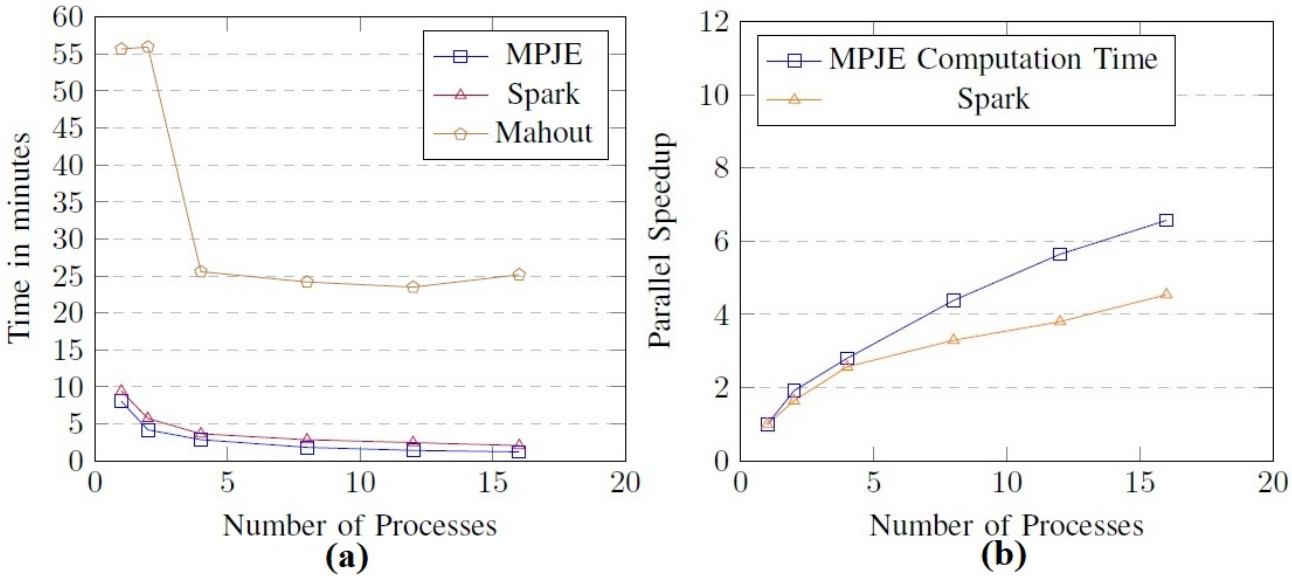


Figure 4: (a) MPJE vs Spark vs Mahout Frameworks performance Comparison of Recommendation System with MovieLens dataset, (b) Parallel Speedup MPJ Express vs Spark with MovieLens dataset

Apache Spark is a cluster computing framework that allows fast parallel computing by supporting multi-core structure[11]. There are other similar frameworks such as: Apache Hadoop, Apache Mahout, MPJ express, can implement a parallel recommendation system structure. But for building a movie recommendation system Apache Spark performs best. Apache Spark has the advantage of fast in memory computing. Because of that Matrix Factorization in Spark is way faster. Study found comparison in performance for different Apache frameworks as figure 4. This performance analysis clearly shows that Apache Spark is the better choice to develop a parallel recommendation system[12]. Some deep learning based recommendation techniques are also achieving high accuracy in performance[13].

3 Problem Statement

a The concern of the study was to find a suitable recommendation algorithm for movie recommendation that works in real time.

b The finding of literature review implies that ALS based Matrix Factorization technique, which is a subclass of user based Collaborative Filtering, is the perfect method to solve our problem. The justification of the claim is very straight forward. The content based algorithm analyze only user profile information. Which cannot lead to good movie recommendation algorithm. Because a particular person's movie preference is not necessarily related to his/her profile information. Not all teenagers like similar movies. So Collaborative Filtering is the better chose for our problem. Again as item based CF suffers from cold-start and popularity bias problem, so user based CF is the appropriate choice. Finally to speed up the recommendation process ALS is the appropriate choice, and Apache Spark is the right framework for that.

c As movie streaming services are very demanding and competitive industries now a day,

a good recommendation system is a key component to complete against each other. That is why developing a well performing recommendation algorithm is significant in technological world.

4 Proposed Solution: ALS Matrix Factorization Collaborative Filtering using Apache Spark

The steps to construct the desired recommendation system are as follows:

1 A new user inputs his/her favorite movies, then system create new user-movie interaction samples for the model

2 System retrains ALS model on data with the new inputs

3 System creates movie data for inference

4 System make rating predictions on all movies for that user

5 System outputs top N movie recommendations for that user based on the ranking of movie rating predictions

The ALS algorithm is described as follows:

Step 1 Initialize matrix M by assigning the average rating for that movie as the first row, and small random numbers for the remaining entries.
Step 2 Fix M, Solve U by minimizing the objective function (the sum of squared errors);
Step 3 Fix U, solve M by minimizing the objective function similarly;
Step 4 Repeat Steps 2 and 3 until a stopping criterion is satisfied.

5 Experimental Evaluation

A movie recommendation system is developed by following the proposed algorithm and it's described steps. The experiment is tasted using suitable standard dataset. This section will discuss about the dataset, experiment environment, output, and limitations.

- **Dataset:** For the purpose of training and testing the system I used the MovieLens ml-100K dataset. There 100,000 ratings (1-5) from 943 users on 1682 movies is recorded. Each user has rated at least 20 movies. So it is a sparse matrix dataset. This is a standard dataset for the problem.

- **Environment:** The experiment is conducted on Apache Spark 2.4.4 version environment. A cluster of several machines has been formed to get parallel execution performance boost. The configuration of each node of the cluster is 32 GB disk space, 8 GB RAM, 4 vCPUper node. Increasing node variation is applied to see the performance change with the increase of nodes.

- **Output:** Apache Spark runs the proposed solution in parallel environment. Finally it produces desired recommendation in very short time with very fast execution. Below

```
(Godfather, The (1972),5.0)
(Trainspotting (1996),5.0)
(Dead Man Walking (1995),5.0)
(Star Wars (1977),5.0)
(Swingers (1996),5.0)
(Leaving Las Vegas (1995),5.0)
(Bound (1996),5.0)
(Fargo (1996),5.0)
(Last Supper, The (1995),5.0)
(Private Parts (1997),4.0)
```

Figure 5: Top 10 movie recommendation for user 789 using the ALS Matrix Factorization CF algorithm.

shows a sample output of top 10 movie recommendation for user number 789 from the system.

Time comparison of the system for different node structures is as below. The Best performance is recorded as 6 seconds.

- Average run time with 4 node environment 10.58 seconds
- Average run time with 8 node environment 6.02 seconds

- **Limitation:** A big optimization problem of the discussed approach is to find appropriate number of latent factor. If this part is not selected properly then system may not work very well.

6 Conclusions

Conclusion of the project can be discussed in following sections:

1 This paper focuses on designing large scale Collaborative Filtering based recommendation system using Apache Spark framework. The main objective was use the parallel computing property of Apache Spark to create a real time recommendation system. As the system is able to process 100k data in few seconds and make meaningful recommendation. So the objective is successfully achieved. Also the proposed system supports multiple distributed node structure, which can adapt to distributed cloud based computation.

2 This research did not optimize the number of latent factor use in the system. Future work can address this and make even more accurate recommendation for the users.

References

[1] H. Zarzour, Z. Al-Sharif, M. Al-Ayyoub, and Y. Jararweh; *A new collaborative filtering recommendation algorithm based on dimensionality reduction and clustering techniques*, 2018 9th International Conference on Information and Communication Systems (ICICS), pages 102-10, ISSN 2573-3346, April, 2018.

[2] J. Zhang, Y. Wang, Z. Yuan, and Q. Jin; *Personalized Real-Time Movie Recommendation System: Practical Prototype and Evaluation*, Tsinghua Science and Technology, vol. 25, no. 2, pages 180-191, April 2020.

[3] B. Kupisz and O. Unold; *Collaborative filtering recommendation algorithm based on Hadoop and Spark*, 2015 IEEE International Conference on Industrial Technology (ICIT), Seville, pages. 1510-1514, 2015.

[4] S. Saravanan; *Design of large-scale Content-based recommender system using hadoop MapReduce framework*, 2015 Eighth International Conference on Contemporary Computing (IC3), Noida, pages 302-307, 2015.

[5] Y. Zhou , D. Wilkinson, R. Schreiber, R. Pan; *Large-Scale Parallel Collaborative Filtering for the Netflix Prize*, Fleischer R., Xu J. (eds) Algorithmic Aspects in Information and Management. AAIM 2008 Lecture Notes in Computer Science, vol 5034. Springer, Berlin, Heidelberg. 2008.

[6] K. Atasu, T. P. Parnell, C.e Dünner, M. Vlachos, H. Pozidis ; *High-Performance Recommender System Training Using Co-Clustering on CPU/GPU Clusters*, 2017 46th International Conference on Parallel Processing (ICPP), pages=372-381, 2017.

[7] X. Tu, S. Liu and R. Li; *Improving matrix factorization recommendations for problems in big data*, 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), Beijing, pages 193-197, 2017.

[8] Y. Azar, A. Nisgav, B. Patt-Shamir; *Recommender systems with non-binary grades*, SPAA '11 Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures Pages 245-252, June 04 - 06, 2011.

[9] A. Al-Doulat; *Surprise and Curiosity in A Recommender System*, 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA), Aqaba, pages 1-2, 2018.

[10] S. Jain, A. Grover, P. S. Thakur and S. K. Choudhary; *Trends, problems and solutions of recommender system*, International Conference on Computing, Communication Automation, Noida, pages 955-958, 2015.

[11] G. Wei-wei and L. Feng; *Application Research of Hadoop's Weibo Recommendation System Prototype Based on Customer Dynamic Behavior*, 2018 3rd International Conference on Smart City and Systems Engineering (ICSCSE), Xiamen, China, pages 562-566, 2018.

[12] C. Diedhiou, B. Carpenter, A. Shafi, S. Sarkar, R. Esmeli and R. Gadsdon; *Performance Comparison of a Parallel Recommender Algorithm Across Three Hadoop-Based Frameworks*, 2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), Lyon, France, pages 380-387, 2018.

[13] J. Lund and Y. Ng; *Movie Recommendations Using the Deep Learning Approach*, 2018 IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, pages 47-54, 2018.