A SPEEDY APPROACH: USER-BASED COLLABORATIVE FILTERING WITH MAPREDUCE

Nilay Narlawar, Ila Naresh Patil

Department of Computer, MMCOE, Pune, India
CSE Department, IES College Bhopal (MP), India

ABSTRACT

The conventional collaborative filtering system generates high-quality recommendations by influencing the likings of society of similar users but it has drawbacks as sparse data problem & lack of scalability. A new recommender system is required to deal with the sparse data problem & produce high quality recommendations in large scale mobile environment. In this paper, the described algorithm of recommendation mechanism for mobile commerce is user based collaborative filtering using Hadoop’s MapReduce with Bloom Filter on distributed environment like cloud computing which solves scalability problem in conventional CF system. The cloud/distributed computing has advantages as flexibility, high efficiency & helps to solve quality problem of mobile commence recommendation system. Bloom filters used in MapReduce will help to reduce the intermediate results in map phase which in turn speed up the overall process of recommendation. This research shows how MapReduce can be used to parallelize Collaborative Filtering. It also presents the architecture to enhance the join performance using Bloom filters in the MapReduce framework.

Keywords: Bloom Filter, Collaborative Filtering, Distributed Environment, MapReduce Algorithm, Recommender System.

1. INTRODUCTION

Many clients like to use the Web to discover product details in the form of online reviews. These reviews are given by other clients and specialists. User-given reviews are becoming more prevalent. Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user. The suggestions relate to various decision-making processes, such as
what items to buy, what music to listen to, or what online news to read. “Item” is the general term used to denote what the system recommends to users. A RS normally focuses on a specific type of item (e.g., CDs, or news) and accordingly its design, its graphical user interface, and the core recommendation technique used to generate the recommendations are all customized to provide useful and effective suggestions for that specific type of item [1].

Recommender systems or recommendation systems are a subclass of information filtering system that seek to predict the ‘rating’ or ‘preference’ that a user would give to an item (such as music, books, or movies) or social element (e.g. people or groups) they had not yet considered, using a model built from the characteristics of an item (content-based approaches) or the user's social environment (collaborative filtering approaches) [2, 3, 4].

The aim of a recommender system is often to "help consumers learn about new products and desirable ones among myriad of choices". [5, 6, 7]

2. WORKING OF USER-BASED COLLABORATIVE FILTERING

Step1: 1) Obtain “User History” in rating matrix, which is a table in which row represents user and column represent items. Intersection represents the rating given by user to item. Absence of value represents user has not given rating to item. This problem is referred as sparse scoring, which is handled by replacing the matrix. [1,2]

Step2: 1) Calculate the similarity between users. For that, many similarity measure methods are available. One of the famous method is person correlation coefficient which is benchmark for CF. We use cosine similarity measure method given by

\[
\sin(x,y) = \frac{\sum s \in S_{xy} y_s x_s}{\sqrt{\sum s \in S_{xy} y_s^2} \sqrt{\sum s \in S_{xy} x_s^2}}
\]

2) Finding the nearest neighbors from the similarity calculations.

Step3: 1) The algorithm calculates the item rating i.e. generates another rating matrix intern. For that the rating is calculated by a weighted average of the rating by the neighbors

\[
\bar{y}_{s} = \bar{y}_x + \frac{\sum y \in S_{xy} (y_y - \bar{y}_x) \sin(x,y)}{\sum y \in S_{xy} \sin(x,y)}
\]

Where, \(\bar{y}_x\) is the average rating of user x.

To reduce the highly intensive computing time and computer resources we purpose new method of CF on Hadoop platform.

3. GENERAL MAPREDUCE OVERVIEW

1) It is a distributed implementation model which is proposed by google.com. The following working described its working on Hadoop platform.

2) The MapReduce model is inspired by the Lisp programming language.
3) It is divided into two phases
   • Map Phase
   • Reduce Phase

Map Phase: Map Phase takes a set of key/value pairs and produces a set of key/values pairs. Here, it groups together all intermediate values associated with the same intermediate key I and passes them to the Reduce phase.

Reduce Phase: It accepts the intermediate key I and a set of values for the key. It merges together these values and produces only one value per reduce invocation.

4) In the Hadoop platform, the default input dataset size of one mapper is less than 64MB file. If the file size is larger than 64MB, the platform would split it into a no. of small files which size less than 64MB automatically.

5) For every i/p file, the Hadoop platform initialize a mapper to deal with it where the file’s line no. as the key and the content of the line as the value. In map phase, the user can define process to deal with the i/p key/value and pass the intermediate key/value to the reduce phase. Finally Hadoop platform would kill the corresponding mapper.

4. PROPOSED RESEARCH ARCHITECTURE

Collaborative Filtering Algorithm can be implemented within the MapReduce framework. It is difficult to directly use MapReduce model in computation process of Collaborative Filtering algorithm. The recommendation process for each user is summarized in the Map function i.e. while making recommendation, we save user ID in text files which serves as input to the Map function. The MapReduce framework defines few mappers to handle the user ID files. Fig 1 shows Application Architecture Diagram.
5. PROPOSED ALGORITHM

The proposed algorithm is divided into three phases:

1) Data Partitioning Phase
2) Map Phase
3) Reduce Phase

5.1 Data Partitioning Phase
Here, it separates the UserID into different files, in these files each row store a UserID. These files are as the I/P to the map phase.

5.2 Map Phase
The Hadoop platform, initialize a new mapper if the Datanode has enough response to initialize a mapper. The mapper’s setup builds the rating matrix between user and item which are already filtered by local filter. The mapper reads the UserID file by line no. Take the line no. as the i/p key and contents of the line as the values.

The local filter of Bloom filter randomly selects 50% users by the random function. In the next step, it computes the similarity between this user and other users.

Finally, it identifies the user’s nearest neighbor (by similarity values) and accordingly with equation 2 to calculate his predict rating on items. The Global filter of the Bloom filter works for the accuracy. It compares the two rating matrices and use e.g. threshold value to select the users from them. The algorithm sort the predict rating and store them in recommendation list. The UserID and its corresponding recommendation list as the intermediate key/value, output them to the reduce phase.

5.3 Reduce Phase
The Hadoop platform would generate some reducers implicitly. The reducers collect the UserID and its corresponding recommendation list, sort them to UserID and then o/p them to the HDFS.
6. EXPERIMENTAL ANALYSIS

6.1 Implementation

We have implemented our experiments for CF algorithm on Java platform. As explained earlier in, the Hadoop computer-cluster created on five computers. Here, we refer one of the computers as MainNode & remaining four as DataSetNodes. Each computer is having 4 GB RAM & Intel(R)core(TM) i5 CPU with 2.5GHz speed & Operating System Ubuntu 10.10. Also the software used for the experiments are Hadoop MapReduce framework, Java JDK 1.6, the Mobile device (Android 3.0 & above), wireless Router are additional hardware we have used. The dataset is created by Netflix data set. The list of different movies is maintained in the dataset and more than 10,000 users. The users will define different ratings for each movie, not necessary the same rating. The role of our CF algorithm is to compare the runtime between standalone & Hadoop platform, so that we don’t focus on accuracy. We take 3 copies of sub-datasets with 100 users, 200 users, 500 users & 1000 users. The DataSetNode is also divided into 2 nodes, 3 nodes, 5 nodes.

6.2 Analysis

For the comparative analysis of standalone & Hadoop platform, we have considered average time $t_{avg}$ as the Hadoop platform at current DataSetNode and the data set running time. Here the speedup is an important criterion to measure the efficiency of our algorithm.

The speedup is given by,

$$ Speedup = \frac{t_{avg}}{t_{std}} $$

In our CF algorithm the recommendation is based on the division of each user theoretically, if we consider N nodes the speedup should be N, in other words, ideally the speedup should be linearly related to the number of DataSetNode.
In the figure 6.2.1 we have shown the analytical result in graph which implies, the time taken by simple Hadoop’s MapReduce is more than Hadoop’s MapReduce with Bloom Filter by increase number of Movies on Distributed environment. Also from the graph we can say that increase in number of DataSetNodes, the speedup increases linearly.
From figure 6.2.2 we can observe that, for 100 users, 200 users, 300 users, 400 users, 500 users the speedup is not linearly increase, this is because the data set is too small, thus the Hadoop platform is unable to demonstrate its efficiency[10].

From figure 6.2.2 and 6.2.3 we can observe that, for 100 users, 200 users, 300 users, 400 users, 500 users the speedup is not linearly increase, this is because the data set is too small, thus the Hadoop platform is unable to demonstrate its efficiency [10].

Fig. 6.2.3: Analysis of Speedup Vs 5 DataSetNodes

7. CONCLUSION

As the amount of information in e-commerce and mobile commerce grows explosively, filtering irrelevant information but finding useful contents and reliable sources has gained more importance. Recommender systems have become a classic tool that interlinks users with information content and sources. However, regardless of its success in many application settings, conventional CF encounters a number of limitations which influence its recommendation accuracy.

Bloom filters used in MapReduce will help to reduce the intermediate results in map phase which in turn speed up the overall process of recommendation. This research shows how MapReduce can be used to parallelize Collaborative Filtering. It also presents an architecture to enhance the join performance using Bloom filters in the MapReduce framework.

8. REFERENCES


