RECOMMENDATION SYSTEM USING BLOOM FILTER IN MAPREDUCE

Reena Pagare and Anita Shinde

Department of Computer Engineering, Pune University
M. I. T. College Of Engineering Pune India

ABSTRACT

Many clients like to use the Web to discover product details in the form of online reviews. The reviews are provided by other clients and specialists. Recommender systems provide an important response to the information overload problem as it presents users more practical and personalized information facilities. Collaborative filtering methods are vital component in recommender systems as they generate high-quality recommendations by influencing the likings of society of similar users. The collaborative filtering method has assumption that people having same tastes choose the same items. The conventional collaborative filtering system 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. MapReduce is a programming model which is widely used for large-scale data analysis. The described algorithm of recommendation mechanism for mobile commerce is user based collaborative filtering using MapReduce which reduces scalability problem in conventional CF system. One of the essential operations for the data analysis is join operation. But MapReduce is not very competent to execute the join operation as it always uses all records in the datasets where only small fraction of datasets are applicable for the join operation. This problem can be reduced by applying bloomjoin algorithm. The bloom filters are constructed and used to filter out redundant intermediate records. The proposed algorithm using bloom filter will reduce the number of intermediate results and will improve the join performance.

KEYWORDS

Collaborative Filtering, MapReduce, Hadoop, Recommender System, Recommender Algorithm, Bloom filter

1. INTRODUCTION

Since the amount of information in e-commerce & mobile commerce increases, there is need to filter irrelevant information & discover helpful contents & reliable sources. Recommender system is the standard tool which gives advice about items, products, information or services users might be fond of. Recommendation systems generate a ranked list of items on which a user might be interested. Recommendation systems are constructed for movies, books, communities, news, articles etc. They are intelligent applications to assist users in a decision-making process where they want to choose one item amongst a potentially overwhelming set of alternative products or services [1]. Recommender systems are personalized information filtering technology used to either predict whether a particular user will like a particular item or to identify a set of N items that will be of interest to a certain user. It is not necessary that a review is equally useful to all users. The review system allows users to evaluate a review's support by giving a score that ranges from "not helpful" to "most helpful". If a particular review is read by all users & found helpful then it can be assumed that new user might appreciate it. Controversial reviews are the reviews that have a variety of conflicting rating (ranking). Controversial review has both passionate followers and motivated enemy without clear majority in either group. The Recommender System uses information from user profile & interaction to tell possible items of concern. It is useful to approximate the degree to which specific user will like a specific product. The Recommender systems are useful in predicting the helpfulness of controversial reviews [2]. Recommender systems are a powerful new technology for extracting additional value for a business from its user databases. These systems help users to find items they want to buy from a industry. Recommender systems benefit users by helping them to discover items they like. They help the business by generating more sales. Recommender systems are fastly becoming a crucial tool in E-commerce on the Web [3].

Most commonly used algorithm for personalized recommendation in commercial recommendation system is Collaborative Filtering (CF). The main problem of CF is its scalability means the calculation cost of CF would be high if the dataset is very large. MapReduce has been widely used for large scale calculation job. It is used to process massive amount of data in a reasonable amount of time. The main advantages of MapReduce are straightforward programming interface and high scalability along with refined failure management. But MapReduce has some restrictions while doing join operation on large scale datasets. The main difficulty in join processing in MapReduce is that complete dataset should be processed and sent to nodes. This creates bottleneck in performance particularly when only small fraction of data is applicable for the join. In this paper, we implement bloom join algorithm on the Hadoop, an open-source execution of MapReduce framework, to enhance the join performance. With this method, we can avoid redundant records and decrease communication overhead.

The rest of this paper is organized as follows. Section 2 explains related work in mobile commerce, MapReduce, joins in MapReduce and bloom filter. Section 3 introduces User- based Collaborative Filtering recommendation algorithm on Hadoop platform. Section 4 describes the proposed architecture. Section 5 shows experiment results and examination. Finally Section 6 concludes the paper.

2. Related Work

2.1 Mobile commerce

In [4], mobile commerce overview, its business applications and technical environment for mobile commerce are explained. Different business needs relevant to mobile commerce are described. It also helps business to define benefits from mobile commerce. It guides about how to use mobile devices to connect people with information about products and to improve association between trading partners in the supply chain and between businesses and customers.

[5] It gives an overview of the basics about m-commerce and e commerce and connection between them. It helps the business managers who are not from IT background to realize the key elements & fundamental problems of m-commerce. It also explains how to determine impact of m-commerce on current and future businesses & to discover new business scenarios. It assists businesses to define benefits derived from mobile commerce and describes the types of mobile commerce applications.

M-commerce history, present information and essential metrics to evaluate usefulness of M-Commerce are described. It focuses on easy m-commerce ideas to make mobile sales and on how to build website mobile friendly [6].

2.2 MapReduce

MapReduce is a programming model for expressing distributed computations on huge amounts of data and an execution framework for large-scale data processing on clusters of product servers. It was initially developed by Google and built on well-known principles in parallel and distributed processing.

A MapReduce program consists of two functions: map and reduce. The map function accepts a set of records from input files in the form of simple key-value pairs and constructs a set of intermediate key-value pairs. The values in the intermediate pairs are automatically collected by key and sent to the reduce function. The grouping consists of sort and merge processes. The reduce function accepts an intermediate key and a set of values related to the key, and finally it generates output key-value pairs [7].

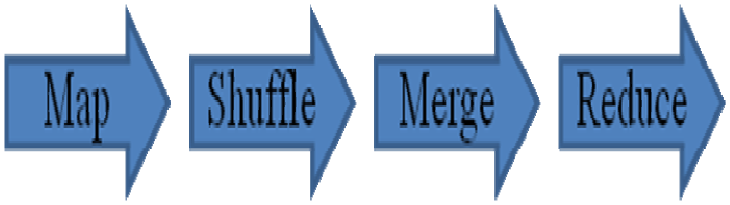

Figure1. Execution Overview of MapReduce

A MapReduce cluster consists of one master node and a number of worker nodes[8]. The master at regular intervals communicates with each worker to test their status and manage their performance. When a MapReduce job is acceptted, the master divides the input data and generates map tasks for the input splits, and reduce tasks. The master allocates each task to idle workers. A map worker accepts the input split and implements map task given by the user. A reduce worker reads the intermediate pairs from all map workers and accomplish reduce task. When all tasks are complete, the MapReduce job is finished.

2.3 Joins in MapReduce

MapReduce’s Join algorithms are generally categorized into two classes: map-side joins and reduce-side joins [9]. Map-side join algorithms are more well-organized than reduce-side joins as only final result of the join is constructed by them in map phase. For Map-Merge join , two input datasets need to be separated and sorted using join key. Whereas reduce-side join algorithms are more common, still they are not competent as the whole input records need to be transferred from map workers to reduce workers[10].

2.4 Bloom filter

A Bloom filter [11] is a probabilistic data structure which is used to check whether an element is a present in a set. It consists of an m-bits array and *k* self-sufficient hash functions. All bits in the array are initialised to 0. When an element is included into the array, the element is hashed *k* times using *k* hash functions, and the location in the array equivalent to the hash values are set to 1. In order to check membership of an element, all bits of its *k* hash positions of the array are examined. If values in all bits are 1, we can say that the element is a member of set.

Bloom-join [12] is a join algorithm which uses the Bloom filter. It makes use of Bloom filter to filter out tuples that are not matched in a join. For example, there are two relations X(a,b) and Y(a,c) situated at location 1 and location 2 respectively. For joining these two relations, Bloom-join creates a Bloom filter with the join key of relation X. Then it forwards the filter to location 2. At location 2, the algorithm scans X and forwards only tuples which are set in the received Bloom filter to location 1. At last, a join of filtered X and Y is executed.

3. User Based Collaborative Filtering Recommendation Algorithm On Hadoop

3.1 Collaborative Filtering

Collaborative filtering algorithm is commonly used recommendation method in business-related recommendation system which improves the performance recommendation process. The main drawback of collaborative filtering is its scalability means when the size dataset is very huge, the cost of calculation would be very high. Cloud computing tries to solve this problem of high calculation task [1]. Collaborative filtering algorithm has assumptions: People have similar likings & concern which are steady. It is possible to guess their option from their past preferences. Collaborative filtering algorithms acquire users' history summary in the form of a ratings matrix consisting of rating given by user to every item. The next step is to compute the similarity between users & discover their nearby neighbours. The most commonly used similarity measure is the Pearson correlation coefficient which is standard for CF.

$$sim(x,y) = \frac{\sum_{s\in S_{xy}} r_{x,s} r_{y,s}}{\sqrt{\sum_{s\in S_{xy}} r_{x,s}^2}\sqrt{\sum_{s\in S_{xy}} r_{y,s}^2}} \qquad (1)$$

Where *rx* is rating of user x on item s and *ry* is rating of user y on item s, *Sxy* indicates the items that user x and y co-evaluated.

The final step is to determine ratings of items which is average of the ratings by the neighbours

$$r_{x,s} = \overline{r}_x + \frac{\sum_{y\in S_{xy}} (r_{y,s} - \overline{r}_x) sim(x,y)}{\sum_{y\in S_{xy}} sim(x,y)} \qquad (2)$$

The CF algorithm requires rigorous computing and computer resources. The calculation process would be very longer if the dataset is very huge.

The job in collaborative filtering is to guess the usefulness of product to a particular user which is based on a database of user votes. Collaborative filtering algorithms guess ranking of a target item for target user with help of grouping of the ranking of the neighbours (similar users) that are known to item under consideration.

3.2 MapReduce and Collaborative Filtering

This part describes how 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 will save user ID in text files which serves as input to the Map function. The MapReduce framework defines few mapper to handle the user ID files. The algorithm is partitioned into three stages as Data segmenting stage, Map stage and Reduce stage.

In Data segmenting stage, the user ID is separated into various files in which every row has a user ID. These files serve as input to the Map stage. This stage needs to fulfil two rules. The first rule is that instead of repeatedly define the mapper, the maximum amount of the execution time should be used up in calculation procedure. The second rule is that each mapper should have same end time.

In Map stage, the Hadoop framework decides whether to define mapper to handle the user ID files. The Hadoop framework defines a new mapper, if sufficient resources are available. Initially the ratings matrix between the consumers and products is constructed by setup function of mapper. Then it gets the line number as the input key and equivalent user ID as value. Then it computes similarity between the user and other users. The last step is to determine the user's nearest neighbours based on similarity values and compute his expected rating on products. These ratings are sorted and stored in recommend list. The user Id and its associated recommend list is given to the Reduce phase.

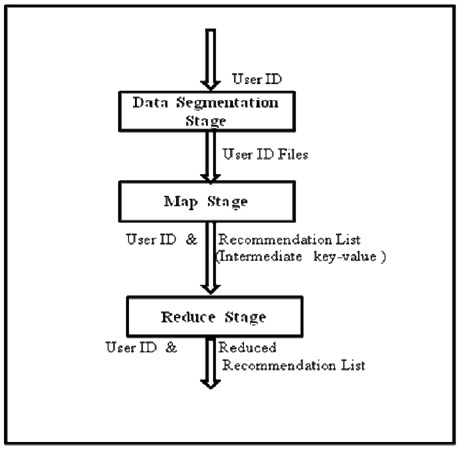
User ID
Data Segmentation Stage
User ID Files
Map Stage
User ID & Recommendation List (Intermediate key-value)
Reduce Stage
User ID & Reduced Recommendation List

Figure 2. Collaborative Filtering Using MapReduce

In Reduce stage, the Hadoop platform automatically creates some reducers which collect the user ID and its associated recommend list, sort them as per their user ID and then forward to the Hadoop Distributed File System(HDFS).

4. PROPOSED ARCHITECTURE

This section illustrates the overall architecture of our implementation. Hadoop is used in our approach, an open-source implementation of the MapReduce framework. In Hadoop, we have master node and worker node. The master node is called jobtracker and the worker node is called tasktracker.

4.1 Execution Overview

Figure 2 shows the overall execution flow of join function on the datasets in our implementation. When the user login to the system by entering username and password, the following sequence of steps is carried out.

1. **Job submission**: When user does login, the calculation of recommendations for active user starts. The users in datasets are getting divided amongst the available tasktracker. Tasktracker contains all the essential information required for map phase.

2. **First Map phase**: The jobtracker gives the map tasks to idle tasktrackers. A map tasktracker accepts the input split for the task, transfers it to key/value pairs, and then executes the map function for the input pairs.

3. **Local filter construction**: Bloom filter is constructed on the key. This filter is called local as they are created only the intermediate results in a tasktracker. Here we get the map output of local filter.

4. **Global filter merging**: All map output of local filter is given to jobtracker which will construct the global filter.

5. **Second Map Phase:** Tasktrackers will execute the given task with received global filter. The Redundant records are filtered out with the help of global filters.

6. **Reduce phase**: A reduce tasktracker examine the resultant intermediate pairs from all map tasktrackers. It arranges the all intermediate pairs and executes the reduce function. At the end, output results are generated from the reduce function.

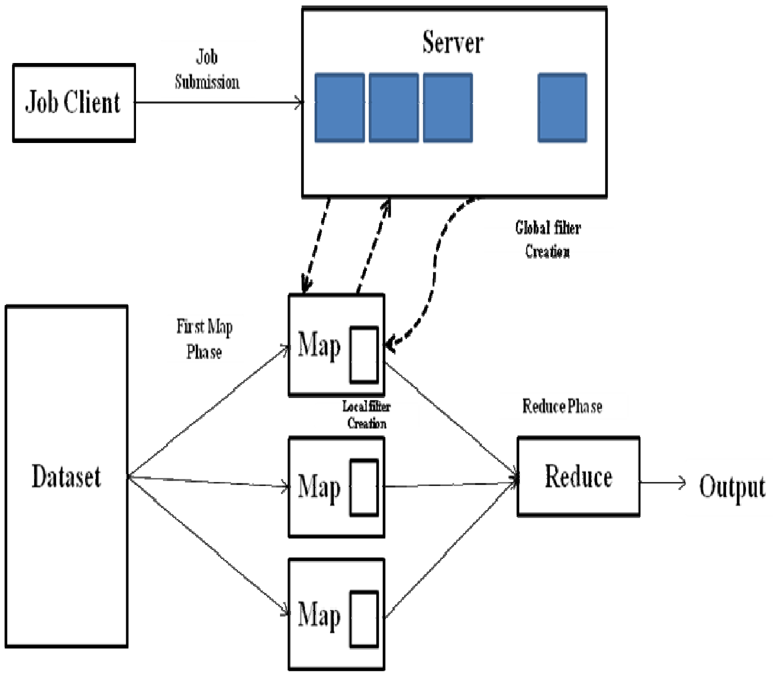

Figure3. Execution Overview

5. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we show our experiment result. We implement User based Collaborative Filtering algorithm on Hadoop platform. Five computers Hadoop cluster was constructed. All experiments were run on this cluster which has one jobtracker (NameNode) and remaining four tasktrackers (DataNodes). Each computer has 4GB RAM & INTEL 2.5GHZ CPU and operating system Ubuntu 10.10. The software used for the experiments includes Hadoop MapReduce framework, Java JDK 1.6. The additional hardware required are the Mobile device (Android 3.0 & above) and switch. In experiments, the Netflix data set is used. The Netflix dataset consists of around 17770 movies and 4, 80,189 users. The main aim of our CF algorithm is to compare the runtime between standalone and Hadoop platform so accuracy is not focussed. We have taken 4 copies of sub-datasets as our experimental datasets which include 100 users, 200users, 500 users and 1000 users. Similarly, DataNode number is considered as 2 nodes, 3 nodes and 5 nodes.

The local filter is constructed on the time interval as months 3, 6,9,12 etc. Then we get the map output of the time interval as we will consider only those reviews of given time interval from specified date. All map output of the local filters is given to job tracker which will construct global filter which will be derived from the movie i.e. How many rating for considered movies. Finally we get reduce operation output. The number of intermediate /output records is constructed from the interval in months (i.e. 3, 6,9,12 etc), map output from reviews and map output from movies. Finally join of reviews and movies will give the reduce output. In this implementation, map phase is terminated early as the intermediate results are decreased by the bloom filters. Similarly the reduce phase is also terminated early as number of intermediate records are decreased, so the number of input records to work with in reduce function are decreased. It is essential to decide the suitable dimension of the Bloom filter. If the dimension of Bloom filter is small, it will be unable to filter out redundant records. Similarly, if the dimension of filters is bulky, it will create large overhead to design and communicate filters.

6. CONCLUSIONS

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 paper 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. We propose to apply this concept to Recommender System for Mobile Commerce.

In future work, we will expand proposed architecture to handle multi-way joins and to determine the appropriate size of Bloom filters.

REFERENCES

[1] Zhi-Dan Zhao, Ming-Sheng Shang. User-Based Collaborative-Filtering Recommendation Algorithmson Hadoop[C]. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, (2010) 478 – 481.

[2] Adomavicius G., Tuzhilin A., Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. on Knowledge and Data Engineering, 2005, 17(6): 734-749

[3] Reena Pagare and Anita Shinde Article : A Study of Recommender System Techniques. International Journal of Computer Applications 47(16):1-4, June 2012. Published by Foundation of Computer Science, New York, USA.

[4] Mobile Commerce: opportunities and challenges A GS1 Mobile Com White Paper February 2008 Edition

[5] Asghar Afshar Jahanshahi, Alireza Mirzaie, Amin Asadollahi MOBILE COMMERCE BEYOND ELECTRONIC COMMERCE: ISSUE AND CHALLENGES Asian Journal of Business and Management SciencesISSN: 2047-2528 Vol. 1 No. 2 [119-129] www.ajbms.org

[6] MOBILE COMMERCE The Future Is Here 3dcart.com/whitepapers ©2010, 3dcart

[7] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In Proceedings of the 6th USENIX Symposium on Opearting Systems Design & Implementation (OSDI), pages 137–150, 2004.

[8] Taewhi Lee, Kisung Kim, and Hyoung-Joo Kim Join Processing Using Bloom Filter in MapReduce RACS'12 October 23-26, 2012, San Antonio, TX, USA. 2012 ACM 978-1-4503-1492-3/12/10 doi>10.1145/2401603.2401626

[9] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon. Parallel data processing with mapreduce: A survey. ACM SIGMOD Record, 40(4):11–20, 2011.

[10] S. Blanas, J. M. Patel, V. Ercegovac, J. Rao, E. J.Shekita, and Y. Tian. A comparison of join algorithms for log processing. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10), pages 975–986, 2010.

[11] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM (CACM), 13(7):422–426, 1970.
[12] L. F. Mackert and G. M. Lohman. R* optimizer validation and performance evaluation for distributed queries. In Proceedings of the 12th International Conference on Very Large Data Bases (VLDB), pages 149–159, 1986.
[13] Zan Huang, Daniel Zeng and Hsinchun Chen A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce 1541-1672/07/$25.00 © 2007 IEEE

AUTHORS

Mrs Reena Pagare has done her graduation and post graduation from Pune University, India in 2001 and 2009 respectively. She is currently working with Maeer's Maharashtra Institute of Technology ,Computer Department, Pune, India.She has 10 yrs of teaching experience. She has published numbers of national and international papers. Her research interest are Recommender Systems, Algorithms.

Anita Shinde has received her BE in 2003 from Pune University and persuing ME from Pune University. She is currently working with MM College of Engineering, Computer Department, Pune, India. She is life member of ISTE. She has published numbers of national and international papers.Her research interest includes Recommender Systems, data mining and information retrieval.