Multithreaded Implementation of the Slope One Algorithm for Collaborative Filtering

Efthalia Karydi and Konstantinos G. Margaritis

University of Macedonia, Department of Applied Informatics
Parallel and Distributed Processing Laboratory
156 Egnatia str., P.O. Box 1591, 54006 Thessaloniki, Greece
`Karydithalia@gmail.com`
`kmarg@uom.gr`

Abstract. Recommender systems are mechanisms that filter information and predict a user's preference to an item. Parallel implementations of recommender systems improve scalability issues and can be applied to internet-based companies having considerable impact on their profits. This paper implements a parallel version of the collaborative filtering algorithm Slope One, which has advantages such as its efficiency and the ability to update data dynamically. The presented version is parallely implemented with the use of the OpenMP API and its performance is evaluated on a multi-core system.

Keywords: Collaborative Filtering, Recommender Systems, Slope One Algorithm, Parallel Programming, OpenMP.

1 Introduction

Collaborative filtering based recommender systems introduce the users' opinion to the procedure of the recommendations generation. Collaborative filtering recommender systems have gained wide popularity. Thus, as the number of users and items of such systems increases, so inevitably does the amount of data. One of the most challenging factors in recommender systems, which is caused due to the data abundance, is the way to achieve high quality recommendations in the shortest time possible. Consequently, a great need is emerging. The achievement of quick data processing, in order to accomplish high quality recommender systems of an increased performance.

In this paper the Slope One algorithm [1] was chosen for parallelization due to the presence of advantages such as its speed and efficacy and the dynamically updatable data. A parallel implementation is introduced using OpenMP and the experimental results are evaluated.

The rest of this paper is organized as follows: In section 2 related work is discussed. Section 3 presents an overview of the different versions of the Slope One algorithm. Section 4 presents the proposed parallel implementation of the Slope One algorithm. The experimental results are analyzed in section 5.

2 Related Work

Recently, the main versions of Slope One are being used together with other algorithms, such as data mining techniques, in order to accomplish faster and more effective recommender systems.

D. Zhang in [2] presented in 2009 a method that used the Slope One algorithm to produce the predictions of a user, and continued by using the Pearson Correlation metric to calculate the neighborhood of similar items and produce recommendations. The quality of the predictions was affected by the size of the set of similar items. This variation was more accurate than the traditional Collaborative Filtering algorithm.

Another method first divides the set of items into subsets, taking into account the kind of items requested by the user. In this manner, the dimension of the set of items, the number of ratings and some times even the number of users, is being reduced. The co-clustering of the data is accomplished by using the K-Means algorithm, taking as parameters the demographic data, that the user must have previously determined. After the dimensionality reduction, Slope One is being used to the smaller dataset to produce the predictions. This approach reduces the time needed for calculations and augments the predictions' accuracy [3].

Between other approaches that use the Slope One algorithm is the one presented in [4], which combines Slope One with Userrank. Userrank is based on Pagerank algorithm and attaches weights to each user, depending on how many related items he has rated. These weights are being used to calculate the differences between the items' ratings. Another algorithm that combines item based with user based collaborative filtering, was proposed in 2009. This algorithm uses Slope One to fill in the empty spaces of the array containing the ratings and on the new dense array applies user based collaborative filtering techniques [5]. Recently, another recommender system based on Slope One has been designed [6]. This approach selects Slope One for being more efficient in the item similarity calculation than other item based algorithms.

The need to accomplish fast real-time recommender systems that are able to handle enormous data, has led the research trends to the study and implementation of parallel collaborative filtering algorithms. [11] presents a parallel algorithm for collaborative filtering, whose purpose is to be scalable to very large datasets. The Alternating Least Squares with Weighted λ Regularization algorithm is implemented using parallel Matlab. [12] presents a parallel collaborative filtering algorithm based on the Concept Decomposition technique, which uses Posix Threads NPTL API on 32 cores and takes 3,2 μs to compute a prediction on Netflix dataset [8].

A distributed algorithm based on co-clustering, which is implemented with MPI and OpenMP, is presented in [13]. The Netflix Prize dataset is used on a 1024-node Blue Gene/P architecture and achieve training time of only 6 seconds and 1,78 μs prediction time per rating. Other parallel co-clustering algorithms exist, as the one described in [14], which simultaneously creates user and item neighborhoods by dividing among the processors the rows and colums of the matrix averages, and the dataflow implementation of a parallel co-clustering algorithm, presented in [15], which uses the Netflix dataset and achieves prediction runtime of 9,7 μs per rating.

Most recent attempts in the field of parallel collaborative filtering algorithms embrace the use of frameworks. To attain better scalability, frameworks such as Hadoop [9] and GraphLab [10] are extensively used. In [16] is implemented a user-based collaborative filtering algorithm on Hadoop, using the Netflix dataset on 9 dual-core processors. Item-based collaborative filtering algorithm is implemented using Hadoop in[17]. This approach seperates the three most excessive computations into four Map-Reduce phases, which are executed in parallel on a three node Hadoop cluster. An open source collaborative filtering library is implemented in [18], using the GraphLab parallel machine learning framework, and two approaches of SGD on Hadoop are presented in [19] and [20].

[21] implements the Weighted Slope One algorithm using Hadoop. This approach clusters users and assigns weights to each cluster. Then, the ratings are predicted using Weighted Slope One. Lately, the research community has drawn further attention to the Slope One algorithm and many approaches have been published [22–25].

3 Background and Notation

In this section is given an overview of the Slope One algorithm. Slope One defines in a pairwise mode, how much better is one item prefered than another by calculating the difference between the items' ratings. One of the main characteristics of the algorithm is that only the ratings of users who have evaluated some common items with the user for whom the prediction is being produced and this user's predictions are introduced in the predictions calculation.

Given a set χ, consisting of all the evaluations in the training set, and two items i and j with ratings u_i and u_j respectively, in a user's u evaluation ($u \in S_{j,i}(\chi)$), the average deviation of u_i regarding u_j is given by

$$dev_{j,i} = \sum_{i \in S_{j,i}(\chi)} \frac{u_j - u_i}{card(S_{j,i}(\chi))} \ . \tag{1}$$

The average deviation of the items is used for the prediction of the rating that the user u would give to item j,

$$pred(u,j) = \bar{u} + \frac{1}{card(R_j)} \sum_{i \in R_j} dev_{j,i} \ , \tag{2}$$

where $R_j = \{i | j \in S(u), i \neq j, card(S_{j,i}(\chi)) > 0\}$ is the set of all relevant items and $card(S_{j,i}(\chi))$ is the number of all the evaluations in the set S that contain ratings for both items i and j.

Two additional versions of the Slope One algorithm exist. The Weighted Slope One, in which the number of observed ratings for each item, $c_{j,i} = card(S_{j,i}(\chi))$, is taken into account and the predictions are calculated according to

$$pred(u,j) = \frac{\sum_{i \in S(u)-\{j\}} (dev_{j,i} + u_i) c_{j,i}}{\sum_{i \in S(u)-\{j\}} c_{j,i}} \, . \tag{3}$$

In the Weighted Slope One version, if a pair of items has been rated by more users than another, this fact affect the predictions.

Another version is the Bi-Polar Slope One, which predicts only if an item will be liked by a user or not. Thus, a two value scale is used instead of a multivalued, which was used in the basic Slope One scheme to predict the exact rating a user would give to an item.

4 Multithreaded Implementation of Slope One

Through the constant increase of data, a need for better processing speed acquisition is emerging. To acomplish better speed, a parallel version of Slope One is implemented using OpenMP, which is described in this section.

The presented version can be applied to shared memory systems. In order for the predictions to be produced, the values of four arrays have to be calculated. The values of the array that contains the ratings that each user has inserted to the system, the values of the arrays which contain the differences and the frequences of the items' appearance in pairs, and finally the values of the deviation matrix. Since these calculations are the most time-consuming part of the code, they are computed in parallel, using the Data Parallel Model. According to this model data is being shared to the threads, and the computations are shared between all threads. The Task Parallel Model has been deliberately avoided, because the calculations needed to the formation of some of the above matrices involve the use of the rest of the matrices. Thus, the use of Task Parallel Model would delay the overall performance. After these calculations, the program calls two functions. One function computes the predictions and the other, the weighted predictions. A parallel region is defined in each of these functions, and each thread produces the predictions for the items that have not been rated.

Pseudocode.

```
Main procedure
main()
{
   1. Initialize OpenMP routines;
   2. All threads compute ratings, differences, frequencies
      and deviation matrices;
   3. For i=0 to users x items, if (ratings[i]==0) then call
      predictions and weighted predictions function;
}
Predictions Function
Predictions()
```

```
{
   1. Calculate prediction of a given user's rating for a given
   item;
   2. Return prediction;
}
Weighted Predictions Function
Weighted predictions()
{
   1. Calculate weighted prediction of a given user's rating for a
   given item;
   2. Return weighted prediction;
}
```

5 Results and Analysis

5.1 Experimental Methodology

The MovieLens dataset, available from GroupLens Research [7], was used for the performance and scalability evaluation of the implementation discused above. MovieLens 100k was used for performance evaluation, and MovieLens 1M was divided into sub-datasets, augmenting the number of users in each one of them and was used for scalability analysis.

The results were compared to those of the sequential algorithm. To achieve this, the OMP_NUM_THREADS environment variable was used to the multithreaded implementation.

The experiments were performed on a system consisted of two CPU's, AMD opteron(tm) Processor 6128 HE, with eight cores each, 800MHz clock speed and 16GB RAM, under Ubuntu Linux 10.04 operating system. The OpenMp version 3.0 was used and time was measured by its omp_get_wtime() function.

5.2 Performance Analysis

As can be seen by the figure 1 the total execution time reduces as the number of used threads increases. With the use of 16 threads, the total execution time is reduced by 9 times over the sequential time. The total execution time measured, refers to the computation time and to both predictions' and weighted predictions' time and their storage to text files. Thus, the total time needed for the predictions' production is less than the total execution time measured in this implementation, because only one of the predictions' functions would be necessary in a recommender system.

The reduction of the total execution time for the MovieLens 100k dataset, in relation to the number of threads used, can be seen in figure 1. About 134 seconds are needed for the sequential execution, and 16 seconds for the multithreaded. Thus, the multithreaded implementation taking advantage of only 16 threads is 8 times faster than the sequential.

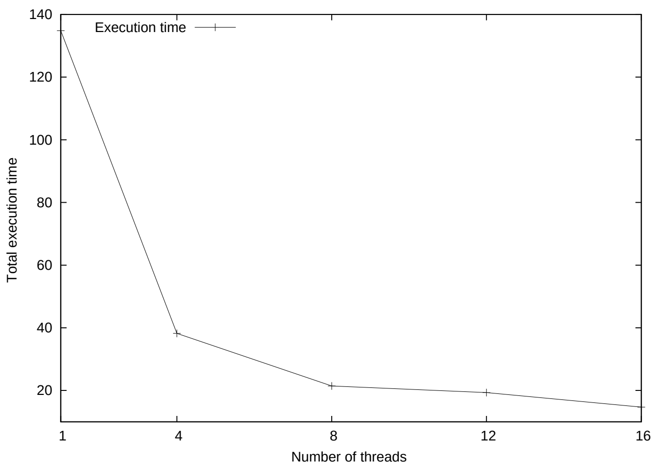

Fig. 1. Total execution time.

Both preprocess time and the time needed to calculate the predictions are reduced. Using 16 threads, the preprocess time in all datasets is reduced in a range from 13.9 to 15.33 times and predictions are generated faster. Both predictions' and weighted predictions' calculations are about 14.5 times faster in the multithreaded implementation. Regarding the number of predictions and weighted predictions produced per second, only 2.2 μs are needed per rating and 1.31 μs per rating using weights.

In figure 2 can be seen the ratio of the sequential implementation's total execution time to the parallel implementation's total execution time on the different datasets. As the dataset size increases, the multithreaded implementation achieves improvement 13.7 times over the sequential implementation.

The number of predictions and weighted predictions per second tends to stabilize to a certain number, as the density of the different datasets remains the same. Figure 3 shows the number of predictions and weighted predictions per second for each dataset. In all datasets after the one containing 248579 ratings, the density ranges between 4.11% and 4.29%. In table 1 can be seen the density of the different datasets and the number of predictions per second on each dataset. In table 2 can be seen some numerical results of the sequential and the multithreaded implementation.

6 Conclusions and Future Work

This paper describes a parallel implementation of the Slope One algorithm and evaluates its performance. Improvement in the execution time, up to 8 times over the sequential execution time, has been achieved. This fact proves that further optimization of the presented approach will not be in vain.

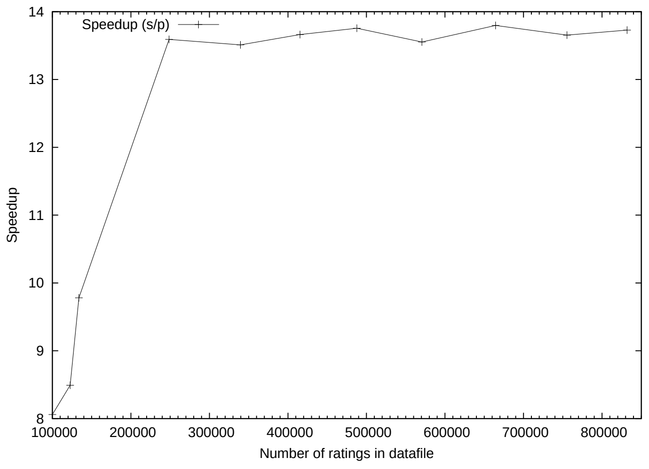

Fig. 2. Speedup on different datasets.

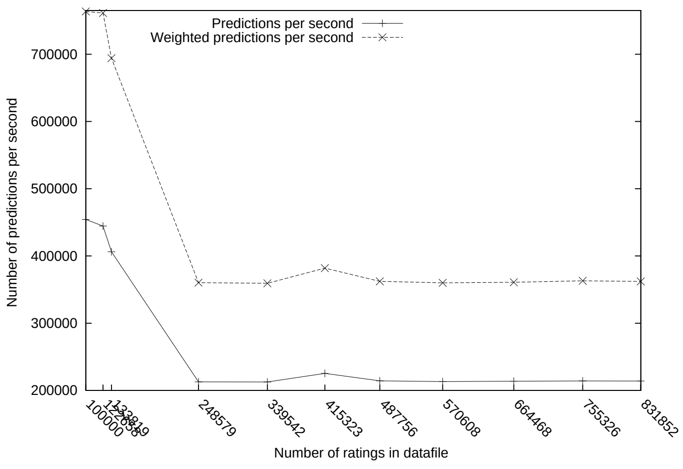

Fig. 3. Predictions and weighted predictions per second.

Table 1. Density of the datasets and predictions per second.

Ratings per dataset	Density %	Predictions per second	Ratings per dataset	Density %	Predictions per second
100000	6.30	454196	487756	4.11	214239
122658	4.54	444371	570608	4.12	213135
133819	4.46	406291	664468	4.20	213732
248579	4.19	212796	755326	4.24	214168
339542	4.29	212621	831852	4.20	214010
415323	4.20	225335			

Table 2. Performance on MovieLens 100k.

Results		
	Sequential	**OpenMP**
Preprocess time	56.88 sec	3.87 sec
Total prediction time	47 sec	3.20 sec
Total weighted prediction time	27 sec	1.90 sec
Predictions per second	31847	454196
Weighted predictions per second	55630	763647
Total time	134.75 sec	16.71 sec
Speedup	—	8.06

In future, optimization techniques will be performed and hybrid approaches will be implemented, in order to improve both execution time and scalability of the multithreaded implementation of Slope One.

References

1. Lemire, D., Maclachlan, A.: Slope One Predictors for Online Rating-Based Collaborative Filtering.SIAM Data Mining (SDM '05), Newport Beach, California (2005)471–476
2. Zhang, D.: An Item-Based Collaborative Filtering Recommendation Algorithm Using Slope One Scheme Smoothing. Second International Symposium on Electronic Commerce and Security ISECS '09 (2009)
3. Mittal, N., Govil, M., Nayak, R. and Jain, K.: Recommender System Framework using Clustering and Collaborative Filtering. Third International Conference on Emerging Trends in Engineering and Technology ICETET (2010) 555–558
4. Gao, M., Wu, Z. and Jiang, F.: Userrank for item-based collaborative filtering recommendation. Information Processing Letters vol. 111 (2011) 440–446
5. Wang, P., Wu Ye, H.: A Personalized Recommendation Algorithm Combining Slope One Scheme and User Based Collaborative Filtering. International Conference on Industrial and Information Systems IIS '09 (2009) 152–154
6. Sun, Z., Luo, N., Kuang, W.: One Real-time Personalized Recommendation Systems Based On Slope One Algorithm. 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD) (2011) 1826–1830
7. GroupLens Research, MovieLens Data Sets, `http://www.grouplens.org/node/73`
8. Netflix Prize, `http://www.netflixprize.com/`
9. Apache Hadoop, `http://hadoop.apache.org/`
10. GraphLab: A New Parallel Framework for Machine Learning, `http://graphlab.org/`
11. Zhou, Y., Wilkinson, D., Schreiber, R., Pan, R.: Large-scale parallel collaborative filtering for the netflix prize. Algorithmic Aspects in Information and Management, Vol.5034 (2008) 337–348
12. Narang, A., Gupta, R., Joshi, A., Garg, V.K.: Highly scalable parallel collaborative filtering algorithm. 2010 International Conference on High Performance Computing (HiPC) (2010) 1–10

13. Narang A., Srivastava A., Katta, N.: Distributed scalable collaborative filtering algorithm. Euro-Par'11 Proceedings of the 17th international conference on Parallel processing. (2011)
14. George T., Merugu S.: A Scalable Collaborative Filtering Framework Based on Co-Clustering. Fifth IEEE International Conference on Data Mining ICDM05(2005) 625–628
15. Daruru S., Marn, N., Walker M., Ghosh, J.: Pervasive Parallelism in Data Mining: Dataflow solution to Co-clustering Large and Sparse Netflix Data. In: KDD '09 Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining ACM Press (2009) 1115–1123
16. Zhao, Z., Shang, M.: User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop. 2010 Third International Conference on Knowledge Discovery and Data Mining (2010) 478–481.
17. Jiang, J., Lu, J., Zhang, G., Long, G.: Scaling-Up Item-Based Collaborative Filtering Recommendation Algorithm Based on Hadoop. World Congress Services (SERVICES) 2011 IEEE (2011) 490 –497
18. Wu Y., Yan Q., Bickson D., Low Y., Yang Q.: Efficient Multicore Collaborative Filtering. In: Matrix (2011)
19. Gemulla R., Nijkamp E., Haas P., Sismanis Y.: Large-scale matrix factorization with distributed stochastic gradient descent. Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining KDD '11 (2011) 69–77
20. Ali M., Johnson C., Tang A.: Parallel Collaborative Filtering for Streaming Data. urlhttp://www.cs.utexas.edu/ cjohnson/ (2011)
21. Chen, X., Hongfa, W.: Clustering Weighted Slope One for distributed parallel computing. Computer Science and Network Technology (ICCSNT) vol.3 (2011) 1595–1598
22. Mi, Z., Xu, C., Huang, D.: A Recommendation Algorithm Combining Clustering Method and Slope One Scheme. Bio-Inspired Computing and Applications. Vol.6840 (2012) 160–167
23. Wang, Y., Yin, L., Cheng, B., Yu, Y.: Learning to Recommend Based on Slope One Strategy. Web Technologies and Applications. LNCS (2012) 537–544
24. Li, J., Sun, L., Wang, J.: A slope one collaborative filtering recommendation algorithm using uncertain neighbors optimizing. WAIM'11 Proceedings of the 2011 international conference on Web-Age Information Management. (2011) 160–166
25. Gao, M., Wu, Z., Luo, Y.: Personalized Context-Aware Collaborative Filtering Based on Neural Network and Slope One. Cooperative Design, Visualization, and Engineering. LNCS vol.5738 (2009) 109–116