The Implementation and Optimization of Matrix Decomposition Based Collaborative Filtering Task on X86 Platform

Tianshu Hao[1] and Ziping Zheng[2,⋆]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
haotianshu@ict.ac.cn
[2] Google, Inc., Mountain View, CA USA
zipingz@google.com

Abstract. With the rapid development of the information age, the recommendation system becomes more and more significant to help people find hidden information from the big dataset in daily lives. Collaborative filtering is a popular technology often used in recommendation systems, which recommend items to users according to other users having the similar behaviors with the target user or according to the items having the alike properties with the target item. In this paper, we implement a parallel collaborative filtering algorithm called ALS-WR on the AMD x86 platform and use an adaptive granularity tuning method to obtain the best performance of 124.86s in 30 training rounds.

Keywords: Recommendation systems · Collaborative filtering · Matrix decomposition.

1 Introduction

With the arrival of the information age, users face a big challenge to extract useful information from a massive amount of data. In addition, it's also difficult for systems to recommend the items to the users who are interested in them. In that, the recommendation system, a kind of information filtering and pushing system, is usually utilized to find the connections between users and items. [1] Collaborative filtering, a key technique in recommendation systems, aims at filling the missing values of the user-item matrix with the help of similar users or items. Collaborative filtering has been widely used by a lot of commercial websites as well as social websites [2], such as Google News [3], Amazon [4], Netflix [5], Reddit [6] and YouTube [7]. The key part of collaborative filtering recommendation system is matrix factorization.

Movie recommendation is a representative application in various recommendation systems. As one of the most widely used movie dataset, MovieLens [8] collects movie rating data from MovieLens website over various periods, describing audiences' preferences for movies. [9]

⋆ Ziping Zheng is the corresponding author

In this paper, we use python to implement the popular Alternating-Least-Square with Weighted-λ-Regularization(ALS-WR) algorithm on AMD x86 platform, and train 30 rounds using MovieLens dataset. Our method achieves a good result on the BenchCouncil International AI System and Algorithm Challenges. Benchcouncil provides four challenges tracks including RISC-V subject[28], Cambricon chip subject[27, 26], X86 platform subject[22, 23] , and 3D face recognition algorithm challenge subject[24, 25]. And these tracks use Benchcouncil AI benchmarks[17, 18, 20, 19, 21]. The source code of AIBench is publicly available from `http://www.benchcouncil.org/benchhub/AIBench/` (Sign up to get access). Eventually, by optimizing and parallelizing the matrix decomposition process, the best training time of 30 rounds of our ALS-WR implementation is 124.86 seconds.

2 Related Work

2.1 ALS

ALS [10] is the abbreviation of alternating least squares, an algorithm always used in recommendation systems based on matrix decomposition. Traditional matrix decomposition Singular Value Decomposition(SVD) is very slow and cannot be used in sparse rating data because it is very common that the users can only rate few items, but ALS solves this problem well in a clever way which using an alternate strategy to update one matrix weight while fixing the another and then vice versa. ALS aims to find two low dimension matrices $X_{m\times k}$ and $Y_{n\times k}$ to approximate the given matrix $R_{m\times n}$.

$$R_{m\times n} \approx X_{m\times k} Y_{n\times k}^T \tag{1}$$

$X_{m\times k}$ calls the user matrix and $Y_{n\times k}$ calls the item matrix. In order to find the user and item matrices, the objective function is:

$$min_{x_u,y_i} L(X,Y) = \sum_{u,i} (r_{ui} - x_u^T y_i)^2 + \lambda(|x_u|^2 + |y_i|^2) \tag{2}$$

with λ being the regularization factor.

2.2 ALS-WR

The model mentioned above is suitable for the scenarios with a clear rating matrix. Nevertheless, users don't give explicit feedback in many cases, which means there is no direct rating. Therefore, we can only infer the preference of the users by their behavior. For example, in the TV recommendation scenario, the recommendation system can speculate the preference of users by analyzing the number of views and the view duration when users watch the TV program. However, there is no way to decide whether users love the TV program when users never watch it. ALS-WR solves above problem by confidence weight: we

assign a larger weight to the item which has the explicit feedback and a smaller weight to the item which doesn't have the explicit feedback.

The objective function of ALS-WR [11] is:

$$min_{x_u,y_i} L(X,Y) = \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(|x_u|^2 + |y_i|^2) \tag{3}$$

$$p_{ui} = \begin{cases} 1 & if\ r_{ui} > 0 \\ 0 & if\ r_{ui} = 0 \end{cases} \tag{4}$$

$$c_{ui} = 1 + \alpha r_{ui} \tag{5}$$

with α being the confidence coefficient.

The solution method of above equation is least squares:

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u r_u \tag{6}$$

$$y_i = (X^T C^i X + \lambda I)^{-1} X^T C^i r_i \tag{7}$$

2.3 OpenBLAS

Basic Linear Algebra Subprograms(BLAS) [12] is a library including a lot of linear algebra operations, such as vector addition, matrix decomposition and so on. Implementations of these operations can be optimized for speed on the upper application to pursue better performance. Considering AMD x86 processors are based on AMD Zen microarchitecture and OpenBLAS [13] has optimized its kernels for Zen-based processors using a template-based methodology [15, 16], we adopt OpenBLAS as our low-level BLAS library to speedup our matrix and vector operations.

3 Implementation and Performance

3.1 Methodology

We present our methodology for implementing collaborative filtering on AMD x86 in figure 1 .

3.2 Implementation

Firstly, we analyze the dataset. The dataset we used is MovieLens 20M Dataset [8], which contains 20 million ratings and 465000 tag applications across 27000 movies by 13800 users. Table 1 shows the main data formats of this dataset.

Secondly, we implement a fast python collaborative filtering using ALS-WR for implicit feedback datasets. [14] Multi-threaded training routines are supported in our implementation, using Python and OpenMP to fit the AMD x86 CPU cores.

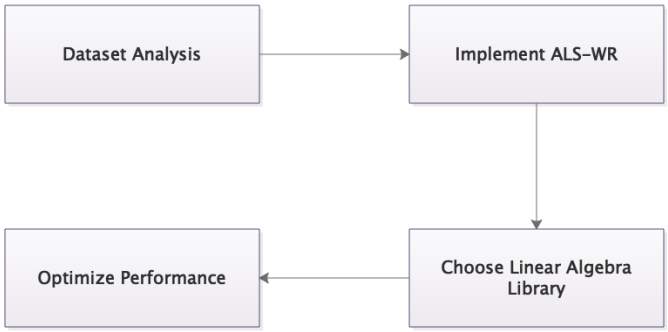

Fig. 1: methodology

Table 1: Data Format of Movielens

File Name	Attribute	Description
movies	moviedID title geners	the attributes of movies
rating	userID movieID rating timestamp	user ratings of the movie
tags	userID movieID tag timestamp	user tags of the movie

Thirdly, we increase the speed of the program by fasting the matrix decomposition, compiling OpenBLAS library on AMD x86 machines.

Finally, we use a mix-grained (fine-grained and coarse-grained) parallel performance tuning framework. We tune both the threads of high-level ALS-WR algorithm and low-level matrix operations, getting the best performance with 256 threads.

3.3 Evaluation

Above all, we implement the ALS-WR algorithm based on OpenBLAS library on AMD x86 CPUs. Figure 2 shows the performance with the different total threads. The best performance reaches 124.86 seconds on 30 rounds using 256 threads.

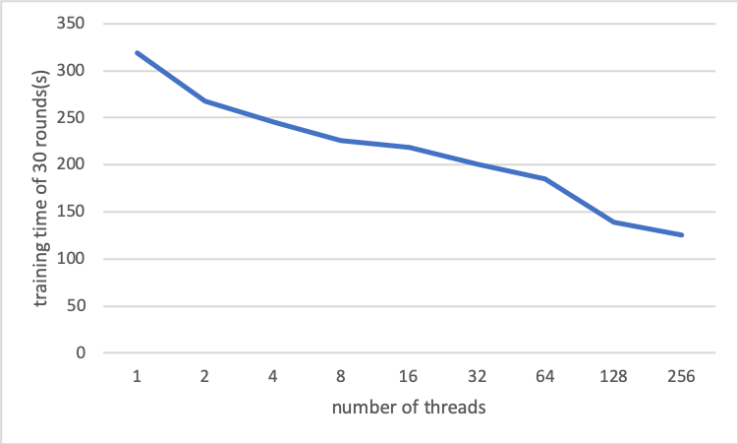

Fig. 2: Performance Chart

4 Conclusion

This paper implements ALS-WR on AMD x86 CPUs. Moreover, we choose OpenBLAS as the underlying linear algebra library to optimize the matrix decomposition. Finally, we use an adaptive granularity tuning method to reach the best training performance (124.86 seconds for 30 rounds).

References

1. Recommender system, `https://en.wikipedia.org/wiki/Recommender\_system`. Last accessed 14 Oct 2019
2. Yang, B., Lei, Y., Liu, J. and Li, W., 2016. Social collaborative filtering by trust. IEEE transactions on pattern analysis and machine intelligence, 39(8), pp.1633-1647.
3. Das AS, Datar M, Garg A, Rajaram S. Google news personalization: scalable online collaborative filtering. InProceedings of the 16th international conference on World Wide Web 2007 May 8 (pp. 271-280). ACM.
4. Linden G, Smith B, York J. Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing. 2003 Jan 1(1):76-80.
5. Netflix, `http://www.netflix.com`. Last accessed 14 Oct 2019
6. Reddit, `https://www.reddit.com`. Last accessed 14 Oct 2019
7. Youtube, `https://www.youtube.com/`. Last accessed 14 Oct 2019
8. MovieLens, `https://grouplens.org/datasets/movielens/`.
9. Harper FM, Konstan JA. The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis). 2016 Jan 7;5(4):19.
10. ALS description in apache flink, `https://ci.apache.org/projects/flink/flink-docs-release-1.2/dev/libs/ml/als.html`. Last accessed 25 Oct 2019
11. Zhou Y, Wilkinson D, Schreiber R, Pan R. Large-scale parallel collaborative filtering for the netflix prize. InInternational conference on algorithmic applications in management 2008 Jun 23 (pp. 337-348). Springer, Berlin, Heidelberg.

12. BLAS, https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms. Last accessed 25 Oct 2019
13. Xianyi Z, Qian W, Saar W. OpenBLAS: An optimized BLAS library. Accedido: Agosto. 2016.
14. Hu Y, Koren Y, Volinsky C. Collaborative filtering for implicit feedback datasets. In2008 Eighth IEEE International Conference on Data Mining 2008 Dec 15 (pp. 263-272). Ieee.
15. Li Z, Jia H, Zhang Y, Chen T, Yuan L, Cao L, Wang X. AutoFFT: a template-based FFT codes auto-generation framework for ARM and X86 CPUs. InProceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis 2019 Nov 17 (p. 25). ACM.
16. Wang Q, Zhang X, Zhang Y, Yi Q. AUGEM: automatically generate high performance dense linear algebra kernels on x86 CPUs. InSC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis 2013 Nov 17 (pp. 1-12). IEEE.
17. Gao W, Tang F, Wang L, Zhan J, Lan C, Luo C, Huang Y, Zheng C, Dai J, Cao Z, Zheng D. AIBench: an industry standard internet service AI benchmark suite. arXiv preprint arXiv:1908.08998. 2019 Aug 13.
18. Gao W, Luo C, Wang L, Xiong X, Chen J, Hao T, Jiang Z, Fan F, Du M, Huang Y, Zhang F. AIBench: towards scalable and comprehensive datacenter AI benchmarking. InInternational Symposium on Benchmarking, Measuring and Optimization 2018 Dec 10 (pp. 3-9). Springer, Cham.
19. Hao T, Huang Y, Wen X, Gao W, Zhang F, Zheng C, Wang L, Ye H, Hwang K, Ren Z, Zhan J. Edge AIBench: towards comprehensive end-to-end edge computing benchmarking. InInternational Symposium on Benchmarking, Measuring and Optimization 2018 Dec 10 (pp. 23-30). Springer, Cham.
20. Jiang Z, Gao W, Wang L, Xiong X, Zhang Y, Wen X, Luo C, Ye H, Lu X, Zhang Y, Feng S. HPC AI500: a benchmark suite for HPC AI systems. InInternational Symposium on Benchmarking, Measuring and Optimization 2018 Dec 10 (pp. 10-22). Springer, Cham.
21. Luo C, Zhang F, Huang C, Xiong X, Chen J, Wang L, Gao W, Ye H, Wu T, Zhou R, Zhan J. AIoT bench: towards comprehensive benchmarking mobile and embedded device intelligence. InInternational Symposium on Benchmarking, Measuring and Optimization 2018 Dec 10 (pp. 31-35). Springer, Cham.
22. Chen M, Chen T,Chen Q. An Efficient Implementation of the ALS-WR Algorithm on x86 CPUs. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
23. Deng W, Wang P, Wang J, Li C, Guo M. PSL: Exploiting Parallelism, Sparsity and Locality to Accelerate Matrix Factorization on x86 Platforms. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
24. Xiong X, Wen X, Huang C. Improving RGB-D face recognition via transfer learning from a pretrained 2D network. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
25. Gong T, Niu H. An Implementation of ResNet on the Classification of RGB-D Images. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
26. Li J, Jiang Z. Performance Analysis of Cambricon MLU100. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.
27. Li G, Wang X, Ma X, Liu L, Feng X. XDN: Towards Efficient Inference of Residual Neural Networks on Cambricon Chips. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.

28. Hou P, Yu J, Miao Y, Tai Y, Wu Y, Zhao C. RVTensor: A light-weight neural network inference framework based on the RISC-V architecture. InInternational Symposium on Benchmarking, Measuring and Optimization 2019(Bench19). Springer.