MapReduce Algorithms for Big Data Analysis

Kyuseok Shim
Seoul National University
shim@ee.snu.ac.kr

ABSTRACT

There is a growing trend of applications that should handle big data. However, analyzing big data is a very challenging problem today. For such applications, the MapReduce framework has recently attracted a lot of attention. Google's MapReduce or its open-source equivalent Hadoop is a powerful tool for building such applications. In this tutorial, we will introduce the MapReduce framework based on Hadoop, discuss how to design efficient MapReduce algorithms and present the state-of-the-art in MapReduce algorithms for data mining, machine learning and similarity joins. The intended audience of this tutorial is professionals who plan to design and develop MapReduce algorithms and researchers who should be aware of the state-of-the-art in MapReduce algorithms available today for big data analysis.

1. INTRODUCTION

There is a growing trend of applications that should handle big data. However, analyzing big data is a very challenging problem today. For such data-intensive applications, the MapReduce [8] framework has recently attracted a lot of attention. MapReduce is a programming model that allows easy development of scalable parallel applications to process big data on large clusters of commodity machines. Google's MapReduce or its open-source equivalent Hadoop [2] is a powerful tool for building such applications.

In the MapReduce framework, a distributed file system (DFS) initially partitions data in multiple machines and data is represented as (key, value) pairs. The computation is carried out using two user defined functions: map and reduce functions. Both *map* and *reduce* functions take a key-value pair as input and may output key-value pairs. The map function defined by a user is first called on different partitions of input data in parallel. The key-value pairs output by each map function are next grouped and merged by each distinct key. Finally, a reduce function is invoked for each distinct key with the list of all values sharing the key. The output of each reduce function is written to a distributed file in the DFS.

The MapReduce framework executes the main function on a single master machine where we may preprocess the input data before map functions are called or postprocess the output of reduce functions. Depending on the applications, a pair of map and reduce functions may be executed once or multiple times.

The research area of developing MapReduce algorithms for analyzing big data has recently received a lot of attentions. In this tutorial, we introduce the MapReduce framework based on Hadoop, discuss how to design efficient MapReduce algorithms and present the state-of-the-art algorithms using MapReduce for big data analysis. The algorithms to be covered are data mining, machine learning and similarity join algorithms.

Proceedings of the VLDB Endowment, Vol. 5, No. 12

2. TUTORIAL OUTLINE

2.1 MapReduce Framework

We start our tutorial by introducing the MapReduce framework including the syntax of map and reduce functions. We provide simple examples of MapReduce algorithms for word counting and building inverted indexes. We also study how to use a *combine* function in MapReduce framework which can improve the performance of MapReduce algorithms significantly. We next discuss how to design efficient MapReduce algorithms and present advanced programming skills that are basic building blocks of designing efficient MapReduce algorithms. These skills include forcing key distributions to reduce functions, broadcasting to mappers and reducers[3, 14, 15], overriding group operators[3, 14, 23] and sharding[7, 14].

2.2 Data Mining

We next focus on MapReduce algorithms including clustering, frequent pattern mining, classification, probabilistic modeling and graph analysis in the context of data mining.

We first practice how to parallelize well-known data mining algorithms such as K-means, EM and CLIQUE[1]. In addition, we discuss parallelization of pre-clustering called Canopy clustering[17]. We then cover MapReduce algorithms for hierarchical clustering[22], density-based clustering[11] and co-clustering[9, 21].

We next focus on parallelization of frequent pattern mining[16] and classification with tree model learning[20]. Furthermore, parallel graph mining algorithms in [6, 12, 13] are studied. Finally, we show how EM algorithms for learning probabilistic model parameters can be parallelized using

MapReduce. The covered parallel algorithms include Probabilistic Latent Semantic Index (PLSI)[7], TWITOBI[14], Latent Dirichlet Allocation (LDA)[24, 25] and Hidden Markov model[5].

2.3 Similarity Joins

We provide an overview of the state-of-the-art in parallel similarity join algorithms which include [3, 10, 15, 18, 23]. While set data is considered in [18, 23], vector data is considered in [3, 10, 15]. The similarity measures for joins considered include Jaccard similarity[18, 23], Ruzicka similarity[18], Cosine similarity[3, 10, 18] and Minkowski distance (i.e. L_p-distance)[15]. The top-k similarity join algorithms using MapReduce are also proposed in [15].

Existing algorithms can be classified into *horizontal partitioning* and *vertical partitioning* methods. Horizontal partitioning algorithms first split all pairs of data points into partitions, next perform similarity joins in each partition separately and finally merge the join results from all partitions. Some pruning techniques are proposed to distribute only candidate pairs (i.e., not all pairs) into partitions[15]. Vertical partitioning algorithms first build inverted indexes and next enumerate every pair to check its similarity in each inverted index separately[3, 10, 18, 23]. When merging the join results from all inverted indexes, we may need *de-duplication* to generate distinct similar pairs only.

In addition to similarity joins, there are interesting studies on developing MapReduce algorithms for other types of joins such as equi-join[4] and θ-join[19]. We give a brief summary of such studies on parallelizing other types of joins too.

3. THE GOAL OF THE TUTORIAL

This tutorial is aimed to offer researchers and practitioners an insight into developing MapReduce algorithms as well as a survey of the current state-of-the-art in MapReduce algorithms for big data analysis. The intended audience of this tutorial is professionals who plan to design and develop MapReduce algorithms and researchers who should be aware of the state-of-the-art in MapReduce algorithms available today for big data analysis.

4. BIOGRAPHY OF THE INSTRUCTOR

Kyuseok Shim received the BS degree in electrical engineering from Seoul National University in 1986, and the MS and PhD degrees in computer science from the University of Maryland, College Park, in 1988 and 1993, respectively. He is currently a professor at Seoul National University, Korea. Before that, he was an assistant professor at KAIST and a member of technical staff for the Serendip Data Mining Project at Bell Laboratories. He was also a member of the Quest Data Mining Project at the IBM Almaden Research Center. He has been working in the area of databases focusing on data mining, cloud computing, recommendation systems, privacy preservation, internet search engines, query processing, query optimization, histograms and XML. His writings have appeared in a number of professional conferences and journals including ACM, VLDB and IEEE publications. He served previously on the editorial board of the VLDB and TKDE Journals. He also served as a PC member for SIGKDD, SIGMOD, ICDE, ICDM, PAKDD, VLDB, and WWW conferences.

5. REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, 1998.

[2] Apache. Apache Hadoop. http://hadoop.apache.org, 2010.

[3] R. Baraglia, G. D. F. Morales, and C. Lucchese. Document similarity self-join with MapReduce. In *ICDM*, 2010.

[4] S. Blanas, J. M. Patel, V. Ercegovac, J. Rao, E. J. Shekita, and Y. Tian. A comparison of join algorithms for log processing in MapReduce. In *SIGMOD*, 2010.

[5] H. Cao, D. Jiang, J. Pei, E. Chen, and H. Li. Towards context-aware search by learning a very large variable length hidden markov model from search logs. In *WWW*, 2009.

[6] J. F. L.-W. H. Y.-M. W. Chao Liu, Hung-chih Yang. Distributed nonnegative matrix factorization for web-scale dyadic data analysis on MapReduce. In *WWW*, 2010.

[7] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW*, 2007.

[8] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI*, 2004.

[9] M. Deodhar, C. Jones, and J. Ghosh. Parallel simultaneous co-clustering and learning with Map-Reduce. In *GrC*, 2000.

[10] T. Elsayed, J. Lin, , and D. W. Oard. Pairwise document similarity in large collections with MapReduce. In *HLT*, 2008.

[11] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan. Mr-dbscan: An efficient parallel density-based clustering algorithm using MapReduce. In *ICPADS*, 2011.

[12] U. Kang, B. Meeder, and C. Faloutsos. Spectral analysis for billion-scale graphs: Discoveries and implementation. In *PAKDD*, 2011.

[13] U. Kang, C. E. Tsourakakis, and C. Faloutsos. PEGASUS: mining peta-scale graphs. *Knowledge and Infomation Systems*, 27(2), 2011.

[14] Y. Kim and K. Shim. TWITOBI: A recommendation system for twitter using probabilistic modeling. In *ICDM*, 2011.

[15] Y. Kim and K. Shim. Parallel top-k similarity join algorithms using MapReduce. In *ICDE*, 2012.

[16] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Chang. PFP: Parallel FP-Growth for query recommendation. *ACM Recommender Systems*, 2008.

[17] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, 2000.

[18] A. Metwally and C. Faloutsos. V-SMART-Join: A scalable MapReduce framework for all-pair similarity joins of multisets and vectors. In *VLDB*, 2012.

[19] A. Okcan and M. Riedewald. Processing theta-joins using MapReduce. In *SIGMOD*, 2011.

[20] B. Panda, J. S. Herbach, S. Basu, and R. J. Bayardo. Planet: Massively parallel learning of tree ensembles with MapReduce. In *VLDB*, 2012.

[21] S. Papadimitriou and J. Sun. DisCo: Distributed co-clustering with Map-Reduce: A case study towards petabyte-scale end-to-end mining. In *ICDM*, 2008.

[22] T. Sun, C. Shuy, F. Liy, H. Yuy, L. Ma, and Y. Fang. An efficient hierarchical clustering method for large datasets with Map-Reduce. In *PDCAT*, 2009.

[23] R. Vernica, M. J. Carey, and C. Li. Efficient parallel set-similarity joins using MapReduce. In *SIGMOD*, 2010.

[24] Y. Wang, H. Bai, M. Stanton, W.-Y. Chen, and E. Y. Chang. PLDA: Parallel latent dirichlet allocation for large-scale applications. In *AAIM*, 2009.

[25] K. Zhai, J. L. Boyd-Graber, N. Asadi, and M. L. Alkhouja. Mr. LDA: A flexible large scale topic modeling package using variational inference in MapReduce. In *WWW*, 2012.