MapReduce Algorithms for Big Data Analysis*

Kyuseok Shim

Seoul National University
Seoul, Korea
shim@ee.snu.ac.kr

Abstract. As there is an increasing trend of applications being expected to deal with big data that usually do not fit in the main memory of a single machine, analyzing big data is a challenging problem today. For such data-intensive applications, the MapReduce framework has recently attracted considerable attention and started to be investigated as a cost effective option to implement scalable parallel algorithms for big data analysis which can handle petabytes of data for millions of users. 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 is a powerful tool for building such applications.

In this tutorial, we will introduce the MapReduce framework based on Hadoop and present the state-of-the-art in MapReduce algorithms for query processing, data analysis and data mining. 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

As there is an increasing trend of applications being expected to deal with big data that usually do not fit in the main memory of a single machine, analyzing big data is a challenging problem today. Examples include the applications for data mining, machine learning and similarity joins. For such data-intensive applications, the MapReduce [1] framework has recently attracted a lot of attention and started to be investigated as a cost effective option to implement scalable parallel algorithms for big data analysis which can handle petabytes of data for millions of users. 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

* A full version of this tutorial was previously presented in VLDB 2012.

A. Madaan, S. Kikuchi, and S. Bhalla (Eds.): DNIS 2013, LNCS 7813, pp. 44–48, 2013.

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 for each distinct key by the shuffling phase. 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. Presentation of a MapReduce algorithm consists of three functions which are map, reduce and main functions. The main function is executed on a single master machine by the MapReduce framework and a pair of map and reduce functions may be executed once or several times.

The research area of developing MapReduce algorithms for analyzing big data has recently received considerable attention. In this tutorial, we introduce the MapReduce framework based on Hadoop, and present the state-of-the-art algorithms using MapReduce for big data analysis. The algorithms to be covered are join processing, data analysis and data mining algorithms.

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 next 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.

2.2 Join Processing

The problem of parallel algorithms for theta joins and similarity joins using MapReduce has been studied in several research communities for various applications. We provide an overview of the state-of-the-art in parallel join algorithms which include [3–9]. The parallel algorithms for the traditional theta join algorithms are introduced in [3, 4] and efficient algorithms for processing n-way theta joins using MapReduce are investigated in [10–12]. For similarity joins, set and vector data are considered in [7, 9] and [5, 6, 8] respectively. The similarity measures for joins considered include Jaccard similarity[7, 9], Ruzicka similarity[7], Cosine similarity[5, 7, 8] and Minkowski distance (i.e. L_p-distance)[6]. The top-k similarity join algorithms using MapReduce are also presented in [6].

2.3 Data Mining

We first practice how to parallelize well-known data mining algorithms such as K-means and EM clustering algorithms. We then cover MapReduce algorithms for hierarchical clustering[13], density-based clustering[14] and co-clustering[15, 16]. We next study parallelization of frequent pattern mining[17] and classification with tree model learning[18]. Furthermore, parallel graph mining algorithms

in [19–21] are also 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)[22], TWITOBI[23], Latent Dirichlet Allocation (LDA)[24, 25] and Hidden Markov model[26].

2.4 Potpourri

We investigate parallel wavelet construction algorithms[27] and nonnegative matrix factorization algorithms[19]. We next cover the MapReduce algorithms for counting triangles in a given graph[28]. In addition, optimizing general MapReduce programs[29, 30] is studied.

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, ICDT, EDBT, PAKDD, VLDB, and WWW conferences.

Acknowledgements. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012-0000111). This research was also supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012033342).

References

[1] Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. In: OSDI (2004)
[2] Apache: Apache Hadoop (2010), http://hadoop.apache.org
[3] Blanas, S., Patel, J.M., Ercegovac, V., Rao, J., Shekita, E.J., Tian, Y.: A comparison of join algorithms for log processing in MapReduce. In: SIGMOD (2010)
[4] Okcan, A., Riedewald, M.: Processing theta-joins using MapReduce. In: SIGMOD (2011)
[5] Baraglia, R., Morales, G.D.F., Lucchese, C.: Document similarity self-join with MapReduce. In: ICDM (2010)
[6] Kim, Y., Shim, K.: Parallel top-k similarity join algorithms using MapReduce. In: ICDE (2012)
[7] Metwally, A., Faloutsos, C.: V-SMART-Join: A scalable MapReduce framework for all-pair similarity joins of multisets and vectors. In: VLDB (2012)
[8] Elsayed, T., Lin, J., Oard, D.W.: Pairwise document similarity in large collections with MapReduce. In: HLT (2008)
[9] Vernica, R., Carey, M.J., Li, C.: Efficient parallel set-similarity joins using MapReduce. In: SIGMOD (2010)
[10] Afrati, F., Ullman, J.D.: Optimizing joins in a Map-Reduce environment. In: VLDB (2009)
[11] Chen, L., Zhang, X., Wang, M.: Efficient multiwaytheta join processing using mapreduce. VLDB (2012)
[12] Wu, S., Li, F., Mehrotra, S., Ooi, B.C.: Query optimization for massively parallel data processin. In: SOCC (2011)
[13] Sun, T., Shuy, C., Liy, F., Yuy, H., Ma, L., Fang, Y.: An efficient hierarchical clustering method for large datasets with Map-Reduce. In: PDCAT (2009)
[14] He, Y., Tan, H., Luo, W., Mao, H., Ma, D., Feng, S., Fan, J.: Mr-dbscan: An efficient parallel density-based clustering algorithm using MapReduce. In: ICPADS (2011)
[15] Deodhar, M., Jones, C., Ghosh, J.: Parallel simultaneous co-clustering and learning with Map-Reduce. In: GrC (2000)
[16] Papadimitriou, S., Sun, J.: DisCo: Distributed co-clustering with Map-Reduce: A case study towards petabyte-scale end-to-end mining. In: ICDM (2008)
[17] Li, H., Wang, Y., Zhang, D., Zhang, M., Chang, E.: PFP: Parallel FP-Growth for query recommendation. ACM Recommender Systems (2008)
[18] Panda, B., Herbach, J.S., Basu, S., Bayardo, R.J.: Planet: Massively parallel learning of tree ensembles with MapReduce. In: VLDB (2012)
[19] Liu, C.,Yang, H.-C., J.F.L.W.H.Y.M.W.: Distributed nonnegative matrix factorization for web-scale dyadic data analysis on MapReduce. In: WWW. (2010)
[20] Kang, U., Meeder, B., Faloutsos, C.: Spectral Analysis for Billion-Scale Graphs: Discoveries and Implementation. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011, Part II. LNCS, vol. 6635, pp. 13–25. Springer, Heidelberg (2011)
[21] Kang, U., Tsourakakis, C.E., Faloutsos, C.: PEGASUS: mining peta-scale graphs. Knowledge and Infomation Systems 27(2) (2011)
[22] Das, A., Datar, M., Garg, A., Rajaram, S.: Google news personalization: scalable online collaborative filtering. In: WWW (2007)
[23] Kim, Y., Shim, K.: TWITOBI: A recommendation system for twitter using probabilistic modeling. In: ICDM (2011)

[24] Wang, Y., Bai, H., Stanton, M., Chen, W.-Y., Chang, E.Y.: PLDA: Parallel Latent Dirichlet Allocation for Large-Scale Applications. In: Goldberg, A.V., Zhou, Y. (eds.) AAIM 2009. LNCS, vol. 5564, pp. 301–314. Springer, Heidelberg (2009)
[25] Zhai, K., Boyd-Graber, J.L., Asadi, N., Alkhouja, M.L.: Mr. LDA: A flexible large scale topic modeling package using variational inference in MapReduce. In: WWW (2012)
[26] Cao, H., Jiang, D., Pei, J., Chen, E., Li, H.: Towards context-aware search by learning a very large variable length hidden markov model from search logs. In: WWW (2009)
[27] Jestes, J., Yi, K., Li, F.: Building wavelet histograms on large data in mapreduce. In: VLDB (2012)
[28] Siddharth Suri, S.V.: Counting triangles and the curse of the last reducer. In: WWW, pp. 607–614 (2011)
[29] Babu, S.: Towards automatic optimization of mapreduce programs. In: SOCC (2010)
[30] Jahani, E., Cafarella, M.J., Re, C.: Automatic optimization for mapreduce programs. In: VLDB (2011)