© 2018 IJSRST | Volume 4 | Issue 2 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X
Themed Section: Scienceand Technology

On Traffic-Aware Partition and Aggregation in Mapreducefor Big Data Applications

[1]Shaik Inthiyaz , [2]S. G. Nawaz, [3]Dr. R. Ramachandra

[1]M.Tech Scholar, Department of Computer Science & Engineering, SKD Engineering College, Gooty, Anantapur, Andhra Pradesh, India

[2]M.Tech, HOD of CSE Department, SKD Engineering College, Gooty, Anantapur, Andhra Pradesh, India

[3]Principal & Professor, Department of Computer Science & Engineering, SKD Engineering College, Gooty, Anantapur, Andhra Pradesh, India

ABSTRACT

The MapReduce programming model simplifies large-scale data processing on commodity cluster by exploiting parallel map tasks and reduce tasks. Although many efforts have been made to improve the performance of MapReduce jobs, they ignore the network traffic generated in the shuffle phase, which plays a critical role in performance enhancement. Traditionally, a hash function is used to partition intermediate data among reduce tasks, which, however, is not traffic-efficient because network topology and data size associated with each key are not taken into consideration. In this paper, we study to reduce network traffic cost for a MapReduce job by designing a novel intermediate data partition scheme. Furthermore, we jointly consider the aggregator placement problem, where each aggregator can reduce merged traffic from multiple map tasks. A decomposition-based distributed algorithm is proposed to deal with the large-scale optimization problem for big data application and an online algorithm is also designed to adjust data partition and aggregation in a dynamic manner. Finally, extensive simulation results demonstrate that our proposals can significantly reduce network traffic cost under both offline and online cases.

Keywords:Map Reduce, Hadoop, Bioinformatics, Cyber Security, Machine Learning, Big Data, Trafficcost

I. INTRODUCTION

Map Reduce has emerged as the most popular computing framework for big data processing due to its simple programming model and automatic managementof parallel execution. Map Reduce and its open source implementation Hadoop have been adopted by leading companies, such as Yahoo!, Google and Face book, for various big data applications, such as machine learning, bioinformatics and cyber security.

Map Reduce divides a computation into two main phases, namely map and reduce, which in turn are carried out by several map tasks and reduce tasks, respectively. In the map phase, map tasks are launched in parallel to convert the original input splits into intermediate data in a form of key/value pairs. These key/value pairs are stored on local machine and organized into multiple data partitions, one per reduce task. In thereduce phase, each reduce task fetches its own share of data partitions from all map tasks to generate the final result. There is a shuffle step between map and reduce phase. In this step, the data produced by the map phase are ordered, partitioned and transferred to the appropriate machines executing the reduce phase.

The resulting network traffic pattern from all map tasks to all reduce tasks can cause a great volume of

net work traffic, imposing a serious constraint on the efficiency of data analytic applications. For example, with tens of thousands of machines, data shuffling accounts for58.6% of the cross-pod traffic and amounts to over 200 petabytes in total in the analysis of SCOPE jobs. For shuffle-heavy Map Reduce tasks, the high traffic could incur considerable performance overhead up to30-40%as shown in. By default, intermediate data are shuffled according to a hash function in Hadoop, which would lead to large network traffic because it ignores network topology and data size associated with each key. As shown in Fig.1, consider a toy example with two map tasks and two reduce tasks, where intermediate dataofthreekeys*K*1, *K*2, and *K*3 are denoted by rectangle bars under each machine. If the hash function assigns data of *K*1 and*K*3 to reducer 1, and *K*2 to reducer 2, a large amount of traffic will go through the top switch. To tackle this problem incurred by the traffic-oblivious partition scheme, take into account of both task locations and data size associated with each key in this paper. By assigning keys with larger data size to reduce tasks closer to map tasks, network traffic can be significantly reduced. In the same example above, if assign *K*1and *K*3 to reducer 2, and *K*2 to reducer1,as shown in

The data transferred through the top switch will be significantly reduced.

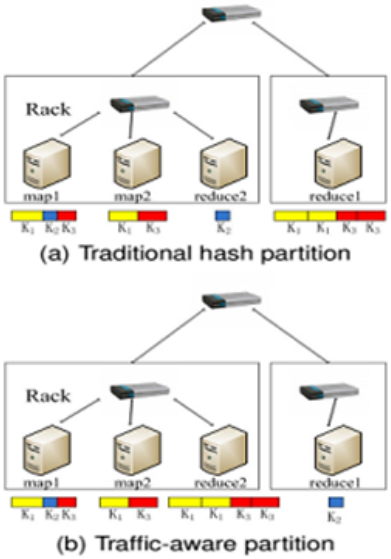

Fig. 1. Two MapReduce partition schemes.

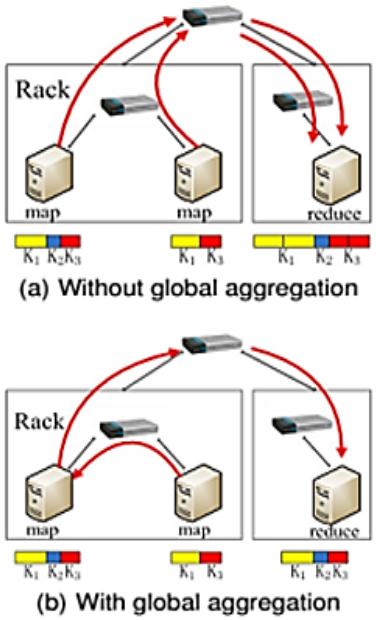

Fig. 2. Two schemes of intermediate data transmission in the shuffle phase.

To further reduce network traffic within a Map Reducejob, consider to aggregate data with the same keys before sending them to remote reduce tasks. Although a similar function, called combiner, has been already adopted by Hadoop, it operates immediately after a map task solely for its generated data, failing to exploit the data aggregation opportunities among multiple tasks on different machines. As an example shown in Fig. 2(a), in the traditional scheme, two map tasks individually send data of key *K*1 to the reduce task. If aggregate the data of the same keys before sending them over the top switch, as shown in Fig. 2(b), the network traffic will be reduced. Jointly consider data partition and aggregation for a Map Reduce job with an objective that is to minimize the total network traffic. Finally, extensive simulation results demonstrate that our proposals can significantly reduce network traffic cost in both offline and online cases.

SYSTEM ANALYSIS

- Domain Analysis
- Requirement Analysis
- Functional Requirements
- Non-Functional Requirements

II. EXISTING SYSTEM

Intermediate data are shuffled according to a hash function in Hadoop, which would lead to large network traffic because it ignores network topology and data size associated with each key. To tackle this problem incurred by the traffic-oblivious partition scheme, take into account of both task locations and data size associated with each key in this paper. By assigning keys with larger data size to reduce tasks closer to map tasks, network traffic can be significantly reduced.

To further reduce network traffic within a Map Reduce job, consider to aggregate data with the same keys before sending them to remote reduce tasks. Although a similar function, called combiner, has been already adopted by Hadoop, it operates immediately after a map task solely for its generated data, failing to exploit the data aggregation opportunities among multiple tasks on different machines.

III. PROPOSED SYSTEM

In the proposed work data partition and aggregation for a Map Reduce job are consider with an objective that is to minimize the total network traffic. In particular, propose a distributed algorithm for big data applications by decomposing the original large-scale problem into several sub problems that can be solved in parallel. Moreover, an online algorithm is designed to deal with the data partition and aggregation in a dynamic manner. Finally, extensive simulation results demonstrate that our proposals can significantly reduce network traffic cost in both offline and online cases.

IV. TEST CASES

TestCase Id	Test Case Name	Test Case Desc	Test Steps			Test Case Status
			Step	Expected	Actual	
Define Reducers 01	Reducer location details	It defines the reducers particular location by providing latitude & longitude values	If we doesn't provide latitude, longitude values	Location details will not be saved	Reducers details will be saved successfully	Fail
Reducer 1 &2 02	Run reducers	Start the reducer nodes ,and all details will be updated at reducer node	If we not run the application	Reducer don't know the updated details	Reducer node will be started	Fail
Upload 03	Upload the input data	Data will be uploaded from shuffle phase	If we can't upload the data	We can't reduce the network traffic	Input data loaded successfully	Fail
Start Mapreduce aggregation 04	Aggregation using Mapreduce	It aggregates all the partitioned data	If we not start the aggregation	We can't reduce the network traffic	After processing the aggregate data, it displays the count result.	Fail

Graph 05	Network traffic cost graph	Displays the graph between processing time & Technique	If we can't do any aggragation	Nothing will be displayed	Graph will be displayed using aggregated/no aggregated data	Fail

Failure Test Case

TestCase Id	Test Case Name	Test Case Desc	Test Steps			Test Case Status
			Step	Expected	Actual	
Define Reducers 01	Reducer location details	It defines the reducers particular location by providing latitude & longitude values	If we doesn't provide latitude, longitude values	Location details will not be saved	Reducers details will be saved successfully	Pass
Reducer 1 &2 02	Run reducers	Start the reducer nodes ,and all details will be updated at reducer node	If we not run the application	Reducer don't know the updated details	Reducer node will be started	Pass
Upload 03	Upload the input data	Data will be uploaded from shuffle phase	If we can't upload the data	We can't reduce the network traffic	Input data loaded successfully	Pass
Start Mapreduce aggregation 04	Aggregation using Mapreduce	It aggregates all the partitioned data	If we not start the aggregation	We can't reduce the network traffic	After processing the aggregate data, it displays the count result.	Pass
Graph 05	Network traffic cost graph	Displays the graph between processing time & Technique	If we can't do any aggragation	Nothing will be displayed	Graph will be displayed using aggregated/no aggregated data	Pass

V. CONCLUSION

Optimization of intermediate data partition and aggregation in MapReducetominimize network traffic cost for big data applications. Propose a three-layer model for this problem and formulate it as a mixed-integer nonlinear problem, which isthen transferred into a linear form that can be solved by mathematical tools. To deal with the large-scale formulation due to big data, we design distributed algorithm to solve the problem on multiple machines. Furthermore, extend our algorithm to handle the MapReduce job inan online manner when some system parameters are notgiven. Finally, we conduct extensive simulations to evaluate our proposed algorithm under both offline casesand online cases. The simulation results demonstratethat our proposals can effectively reduce network trafficcost under various network settings.

VI. FUTURE ENHANCEMENT

Furthermore,to extend our algorithm to handle the MapReduce job inan online manner when some system parameters are notgiven. Finally, conduct extensive simulations to evaluate our proposed algorithm under both offline casesand online cases. The simulation results demonstratethat our proposals can effectively reduce network trafficcost under various network settings.

VII. REFERENCES

[1]. J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1,pp. 107-113, 2008.

[2]. W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, "Map taskscheduling in mapreduce with data locality: Throughput andheavy-traffic optimality," in INFOCOM, 2013 Proceedings IEEE.IEEE, 2013, pp. 1609-1617.

[3]. F. Chen, M. Kodialam, and T. Lakshman, "Joint scheduling of processing and shuffle phases in mapreduce systems," in INFOCOM,2012 Proceedings IEEE. IEEE, 2012, pp. 1143-1151.

[4]. Y. Wang, W. Wang, C. Ma, and D. Meng, "Zput: A speedy datauploading approach for the hadoop distributed file system," inCluster Computing (CLUSTER), 2013 IEEE International Conferenceon. IEEE, 2013, pp. 1-5.

[5]. T. White, Hadoop: the definitive guide: the definitive guide. " O'Reilly Media, Inc.", 2009.

[6]. S. Chen and S. W. Schlosser, "Map-reduce meets wider varietiesof applications," Intel Research Pittsburgh, Tech. Rep. IRP-TR-08-05,2008.

[7]. J. Rosen, N. Polyzotis, V. Borkar, Y. Bu, M. J. Carey, M. Weimer,T. Condie, and R. Ramakrishnan, "Iterative mapreduce for largescale machine learning," arXiv preprint arXiv:1303.3517, 2013.

[8]. S. Venkataraman, E. Bodzsar, I. Roy, A. AuYoung, and R. S.Schreiber, "Presto: distributed machine learning and graph processing with sparse matrices," in Proceedings of the 8th ACMEuropean Conference on Computer Systems. ACM, 2013,pp.197-210.

[9]. A. Matsunaga, M. Tsugawa, and J. Fortes, "Cloudblast: Combining mapreduce and virtualization on distributed resources forbioinformatics applications," in eScience, 2008. eScience'08. IEEEFourth International Conference on. IEEE, 2008, pp. 222-229.

[10]. J. Wang, D. Crawl, I. Altintas, K. Tzoumas, and V. Markl, "Comparison of distributed data-parallelization patterns for big dataanalysis: A bioinformatics case study," in Proceedings of the FourthInternational Workshop on Data Intensive Computing in the Clouds(DataCloud), 2013.

[11]. R. Liao, Y. Zhang, J. Guan, and S. Zhou, "Cloudnmf: A mapreduce implementation of nonnegative matrix factorization for largescale biological datasets," Genomics, proteomics & bioinformatics,vol. 12, no. 1, pp. 48-51, 2014.