International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 14
November 2017

A Study Of Traffic Aware Partition And Aggregation In Mapreduce For Big Data Applications

Lade Praveenkumar & Gugulothu Praveen
*Pg Scholar,Department Of Cnis, Vaagdevi College Of Engineering Bollikunta,Warangal,Telangana
** Asst. Professor, Department Of Cse, Vaagdevi College Of Engineering Bollikunta,Warangal,Telangana

ABSTRACT:

MapReduce programming framework process large amount of data by taking advantage of parallel Map and Reduce tasks. Computationally MapReduce has two phases called Map and Reduce. In actual implementation, it has another phase called Shuffle where data transfer takes place. Conventionally Shuffle phase use Hash function to partition data which is inefficient in handling Traffic leading to a bottleneck. Improving the performance of network traffic inshuffle phase is important to improve the performance of MapReduce. The goal of minimization of network traffic is achieved by using partition and aggregation. The proposed scheme is designed to minimize network traffic cost in MapReduce. The problem of aggregator placement is considered, where each aggregator can reduce combined traffic from multiple map tasks.A decomposition-based distributed algorithm is proposed to deal with the large-scale optimization problem for big data applications. Also, an

online algorithm is designed to dynamically adjust data partition and aggregation

INTRODUCTION

Big Data has emerged as a widely recognized trend, attracting attentions from government, industry and academia. Generally speaking, Big Data concerns large-volume, complex, growing data sets with multiple, autonomous sources. The major challenge for the Big Data applications is to process the large volumes of data and extract useful information for future actions. MapReduce has appeared as the very popular calculating framework for big data processing appropriate to its simple programming model and automatic parallel execution. MapReduce and Hadoop have been used by many big companies, such as Yahoo!, Google and Facebook, fordifferent big data applications. In MapReduce [1][2], computation is viewed as consisting of two phases, called `map' and `reduce' respectively. In the map phase, data is reorganized in such a manner that the desired computation can then be achieved by uniformly applying one algorithm on small portions of the data. The second phase in MapReduce is called

the reduce phase. As each of these two phases can achieve massive parallelism, MapReduce systems can exploit the large amount of computing power by huge scale clusters. When understanding the performance of MapReduce systems, it is convenient to view a MapReduce job as consisting of three phases rather than two phases. The additional phase, which is considered between the map phase and the reduce phase, is a data transfer phase called the `shuffle' phase. In the shuffle phase, the output of the map phase is recombined and then transferred to the compute nodes that are scheduled to perform corresponding reduce operations. The performance of MapReduce systems clearlydepends heavily on the scheduling of tasks belonging to thesethree phases. Even though many efforts have been made to improve theperformance of MapReduce jobs, they show blind eye to thenetwork traffic generated in the shuffle phase, which plays acrucial role in performance enhancement. In traditional way, ahash function is used to partition intermediate data amongreduce tasks, which, however, is not traffic-efficient because we don't consider network topology and data size associated with each key. In this paper, by designing a novel intermediate data partition scheme we reduce network traffic cost for a MapReduce job. Configuring the job, submitting it, controlling its execution, and querying the state is allowed to user by Hadoop. Each and every job consists of independent tasks, and all the tasks need to have a system slot to run. All scheduling and allocation decisions in Hadoop are made on a task and node slot level for both the map and reduce phases. The Hadoop scheduling model is a Master/Slave cluster structure. The master node (Job Tracker) coordinates the worker machines (Task Tracker). Job Tracker is a process which manages jobs, and Task Tracker is a process which manages tasks on adjacent nodes. The scheduler resides in the Job tracker and allocates to Task Tracker various resources to running tasks: Map and Reduce tasks are granted independent slots on each machine. MapReduce Scheduling system takes on in six steps: First, User program divides the MapReduce job. Second, master node distributes MapTasks and ReduceTasks to different workers. Third, MapTasks reads in the data splits, and runs mapfunction on the data which is read in. fourth, MapTasks write intermediate results into local disk. Then, ReduceTasks read the intermediate results remotely, and run reduce function on the intermediate results which are read in.

II. RELATED WORK

1. "Map Task Scheduling in MapReduce with Data Locality: Throughput and Heavy-Traffic Optimality" [2] While assigning map tasks, a critical consideration is to place map tasks on or close to machines that store the chunks of input data, a problem is called data locality. For each and every task, we call a machine a local machine for the task if the data chunk associated with the task is stored locally, and this task is called local task on the machine; the machine is called a remote machine for the task and correspondingly this task is called a remote task on the machine. We need to achieve the right balance between data locality and load-balancing in MapReduce algorithm thatallocates map tasks to machines a map-scheduling algorithm or simply a scheduling algorithm is used.

2. "Zput: a speedy data uploading approach for the Hadoop Distributed File System"[4] The main motive of Zput is remapping files in the native file system directly into thenamespace of HDFS, which are disguised as HDFS blocks. To overcome the unbalanced data distribution problem, we implement the mechanism to replicate blocks remotely based on Zput, whose only goal is to achieve a more balanced and efficient distribution for data blocks.

3. "Online aggregation and continuous query support in MapReduce," [8] This extends the MapReduceprogramming model beyond batch processing, and can reduce completion times and improve system utilization for batch jobs as well. A modified version of the Hadoop MapReduce framework that supports online aggregation is demonstrated, which allows users to see "early returns" from a job as it is being computed.

4. "Purlieus: Locality aware resource allocation for mapreduce in a cloud" [10], describe locality awareness during both Map and Reduce phases. Thislocality-awareness during bothmap and reduce phases of thejob not only improves runtimeperformance of individual jobs butalso has an additional advantageof reducing network trafficgenerated.

5. "Camdoop: Exploiting in-network aggregation for big data applications" [12], Camdoop exploits the property that CamCube servers forward traffic to perform in-network aggregation of data during the shuffle phase. Camdoop supports the same functions used in MapReduce and is compatible with existing MapReduce applications. We demonstrate that, in common cases, Camdoop significantly reduces the network traffic and provides high performance increase over a version of Camdoop.

6. "Hadoop acceleration in an open flow-based cluster," present detailed study of how Hadoop

can control its network resources using OpenFlow in order to improve performance.

7. "Comprehensive View of Hadoop MapReduce SchedulingAlgorithms" The Scheduling is one of the most critical aspects of MapReduce because of the important issues that wedescribed, and many more problems in scheduling of MapReduce. To overcome these issues with different techniques and approaches many algorithms are presented and studied. Some of these algorithm focuses toimprovement data locality and some of them implements to provide Synchronization processing.Many of these algorithms have been designed to minimize the total completion time. Some of these algorithms are FIFO scheduling algorithm, Fair Scheduling Algorithm, Capacity scheduler, hybrid scheduler based on dynamic priority, etc. Each has its own advantages and disadvantages.

III. PROPOSED SYSTEM

A. Design Considerations To reduce network traffic within a MapReduce job, we have to consider aggregate data with similar keys before sending them to remote reduce tasks. Even though we have a similar function, called combiner, which 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. Objective is to minimize the total network traffic by Data partition and aggregation for a MapReduce job. Distributed algorithm is proposed for big data applications by decomposing the original large-scale problem into several subproblems and these subproblems can be solved in parallel. Another is online algorithm which is also designed to deal with the data partition and aggregation in a dynamic manner.

B. System Architecture

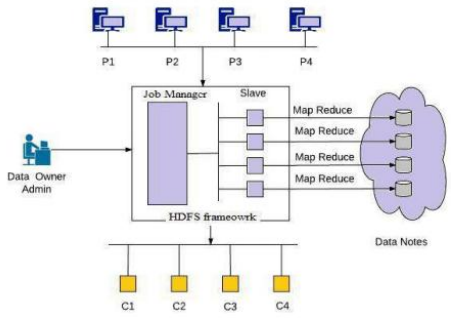

Fig 1. Architecture of Proposed System

The incoming big data from data generators is received by the Job Manager where it is partitioned and Map/Reduce Tasks are carried out. The data is portioned and stored on the nodes by using load balancing techniques to minimize traffic. The Clients ask queries to the system.

IV.CONCLUSION AND FUTURE WORK

The joint optimization of intermediate data

partition and aggregation in MapReduce to minimize network traffic cost for big data applications is studied. The technique of load balancing is used to reduce traffic. Furthermore, we plan to extend our algorithm to handle the MapReduce job in an online manner when some system parameters are not given. Finally, we will conduct extensive simulations to evaluate our proposed algorithm under both offline cases and online cases.

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 task scheduling in mapreduce with data locality: Throughput and heavy-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 data uploading approach for the hadoop distributed file system," in Cluster Computing (CLUSTER), 2013 IEEE International Conference on. 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 varieties of applications," Intel Research Pittsburgh, Tech. Rep. IRP-TR-08-05, 2008.

[7] F. Ahmad, S. Lee, M. Thottethodi, and T. Vijaykumar, "Mapreduce with communication overlap," pp. 608–620, 2013.

[8] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, J. Gerth, J. Talbot, K. Elmeleegy, and R. Sears, "Online aggregation and continuous query support in mapreduce," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010, pp. 1115–1118.

[9] A. Blanca and S. W. Shin, "Optimizing network usage in mapreduce scheduling."

[10] B. Palanisamy, A. Singh, L. Liu, and B. Jain, "Purlieus: localityaware resource allocation for mapreduce in a cloud," in Proceedings of 2011 International Conference for High

Performance Computing, Networking, Storage and Analysis. ACM, 2011, p. 58..

[11] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "Mapreduce online." in NSDI, vol. 10, no. 4, 2010, p. 20.

[12] P. Costa, A. Donnelly, A. I. Rowstron, and G. O'Shea, "Camdoop: Exploiting in-network aggregation for big data applications." in NSDI, vol. 12, 2012, pp. 3–3.