EFFECTIVE REDUCTION OF NETWORK TRAFFIC COST IN MAP REDUCE FOR VERY LARGE SCALE DATA APPLICATIONS

By

K. REDDAMMA * D. JAGADEESAN ** T. VIVEKANANDAN ***

* Research Scholar, Department of Computer Science and Engineering, Sreenivasa Institute of Technology and Management Studies, A.P., India.
** Professor, Department of Computer Science and Engineering, Sreenivasa Institute of Technology and Management Studies, A.P., India.
*** Associate Professor, Department of Computer Science and Engineering, Sreenivasa Institute of Technology and Management Studies, A.P., India.

Date Received: 27/01/2017 Date Revised: 15/03/2017 Date Accepted: 08/05/2017

ABSTRACT

The MapReduce programming model provides an exciting opportunity to process massive volumes of heterogeneous data using map and reduce tasks in parallel. In the recent time, a number of efforts has been made to improve the performance of the job's execution. The performance of the job's execution can be improved further by considering the network traffic. In this paper, an optimistic distributed algorithm is proposed to deal with the significant optimization problem for handling large size data. The optimistic distributed algorithm is more efficient than the distributed algorithm. Finally, simulation results show that the proposal can significantly reduce network traffic cost.

Keywords: Aggregator, Big Data, Hadoop, Hash Table, MapReduce, Optimistic Distributed Algorithm.

INTRODUCTION

In the recent time, the amount of data being generated is rapidly evolving from terabytes to many petabytes. The traditional data processing tools available, fail to handle the large data size. The tools are evolving to overcome the inadequacies existing and provides a solution to manage and process massive volumes of heterogeneous data. The current technologies support to manage large data storage, processing, analyzing, effective decision making, etc. When the size of the data increases the network traffic will also increase as well.

The High Performance Computing (HPC) and Grid computing [1], [2] have been adopted for large scale data processing. The idea implemented in HPC is to work with the cluster of machines and access a shared distributed file system. This is especially used for highly computational intensive jobs. The issue in the above computing approach is network bandwidth.

If the network traffic develops largely, network bandwidth is restricted, and hence computing nodes in the high performance cluster become idle. The HDFS (Hadoop Distributed File System) is a distributed file system constructed to run on commodity hardware. It has many features similar to other distributed computing. HDFS is designed to provide more fault-tolerance and to deploy on low-cost hardware. It is also designed to process and manage massive volumes of heterogeneous data. The Hadoop can store, analyze and process large volumes of data. Also, Hadoop has capacity of large system, fault-tolerance, scalability, and availability.

MapReduce [3] is a software framework used to write applications that process large amounts of data in parallel on clusters of commodity hardware.

A MapReduce job first divides the data into individual chunks which are processed by Map jobs in parallel. The outputs of map sorted by the framework are then input to the reduce tasks. In general, the input and the output of the jobs are both stored in a file system. The MapReduce online is a modified version of Hadoop [4] MapReduce, which supports online aggregation and reduces response time. This approach has the advantage of simple recovery in the case of processing into two phases: (a) Map phase and (b) Reduce phase. A map uses a common partitioner records that is partitioned into hashed buckets [9], [10]. Reduce reads from every mapper for one designated partition, use the same hash function records from these partitions are grouped and aggregated using a hash table.

1. Related Work

Blanca and Shin [4] have proposed "Optimizing network usage in MapReduce Scheduling". In this paper, they introduced a technique called NUM (Network Utilization Monitoring). It maintains a high network utilization and low network congestion. Palaniswamy, Singh, Liu, and Jain [5] have proposed MapReduce resource allocation system. It is used for reducing network traffic generated in the cloud data center.

Costa, Donnelly, Rowstron, and O' Shea [6] have proposed Camdoop, instead of increasing the bandwidth; it focuses on decreasing the network traffic by pushing aggregation from the edge into the network. Camdoop significantly reduces the network traffic and provides high performance increases over a version of Camdoop running over a switch and against two production systems.

Condie, et al., [7] have proposed a modified MapReduce architecture that allows data to be pipelined between operators. Pipeline parallelism is a new option for improving performance of MapReduce jobs, but needs to be integrated intelligently with both intra-task partition parallelism and speculative redundant execution for straggler handling.

In the recent time, IT industry built data-center-scale-computing systems to meet high storage and processing demands of data-intensive applications. Chen and Schlosser [8] in their work, have reported a desirable feature to improve the MapReduce model for data intensive applications. F. Ahmad, et al., [9] have proposed MapReduce with Communication Overlaps (MaRCO) to achieve nearly full overlap via these novel ideas of including reduction in the overlap.

2. Problem Identification

The MapReduce process consumes more bandwidth, and hence it generates the network traffic. In the existing approach, the distributed algorithm is used for traffic aware partition and aggregation is implemented with non-negative values for analysis. In this paper, the authors have proposed an optimized distributed algorithm to solve the above problem and minimize the network traffic.

3. Proposed Architecture

The data intensive task uses data parallelism approach to process massive volumes of data in terms of terabytes and petabytes. The amount of data size is increasing as a result, the network traffic also increases. The data management and analysis of data is time consuming. This may vary for different machines. The conventional data management system fails to handle the enormous volumes of data. The MapReduce programming model in Hadoop framework is designed to process massive volume data using large clusters of commodity hardware that are highly fault tolerant and reliable. The performance of job execution in the MapReduce model can be improved further by considering network traffic.

Figure1 shows the architecture proposed for minimization of network traffic in the MapReduce model. The processes are split into three modules. They are Mapping, Aggregation, and Reduction. First, the user has to define the reducer location for optimizing the data transfer locations (Figure 2). Then the user has to upload the massive volumes of data in HDFS (Hadoop Distributed File System). The aggregation intention is to obtain more information about particular groups based on specific variables, such as location, year, frequently used words, profession, or age. The aggregator is used after completion of mapped jobs and before sending to reducer as the first stage of minimizing network. This aggregator reduces the traffic and finally sents to reducer for the output of the data.

The proposed work "Effective Reduction of Network Traffic Cost in MapReduce for Very Large Scale Data Applications" is implemented in the following phases:

Figure 3 shows the overview of MapReduce program execution. A MapReduce program has three main components, viz., a driver class, a mapper class, and a reducer class (package org.apache.hadoop. mapreduce). The basic driver initializes the job

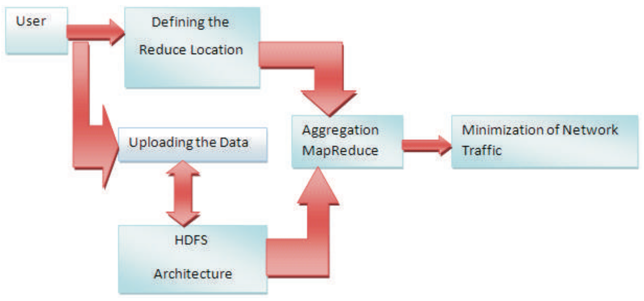
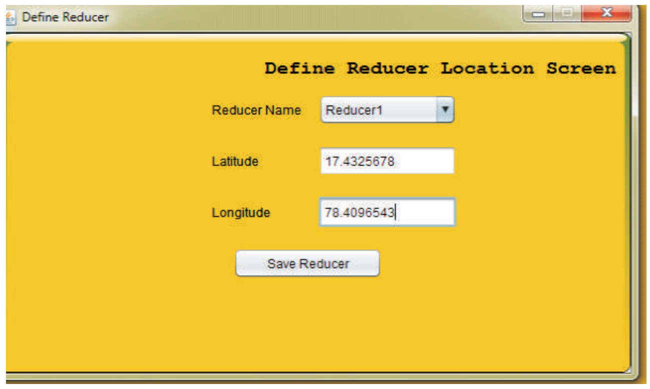


Figure 1. Architecture for Minimization of Network Traffic for MapReduce Model



Define Reducer Location Screen

Reducer Name: Reducer1

Latitude: 17.4325678

Longitude: 78.4096543

Save Reducer

Figure 2. Defining the Reduce Location

configuration, defines the mapper and the reducer and specifies the paths of the input and output files(s) for the MapReduce program.

Figure 4 illustrates the process of MapReduce and Aggregation. Three main functions were used to obtain the optimized data.

3.1 Mapping Function

In mapper function, map tasks are performed in parallel that splits input data into intermediate data in the form of key value pairs. These key value pairs are stored on the local machine and organized into multiple data partitions. The map is a user-defined function, which takes a series of key value pairs and processes, where each one of them to generate zero or more key value pairs.

MAP(): Generates the intermediate key and value based on input key and input value.

3.2 Reducing Function

In reducer function (Figure 5), each reduce task fetches its data as data partitions from all map tasks to generate the final result. The Reducer takes input on the grouped key value paired data and executes a reducer function on each one of them. At this point, the data can be aggregated, filtered, and combined in a number of ways, and it requires an extensive range of processing. Once

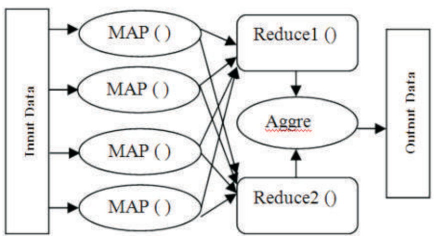


Figure 4. MapReduce Process

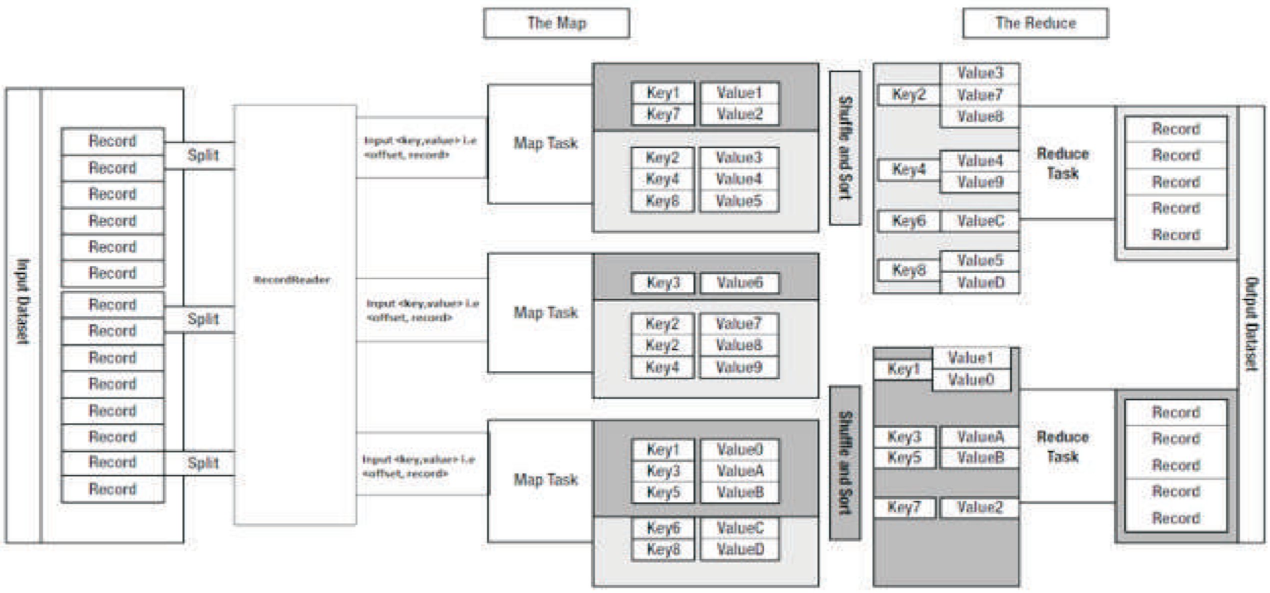
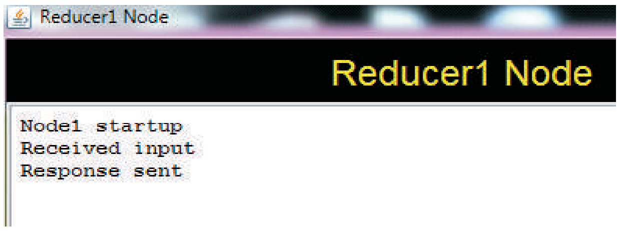


Figure 3. Overview of MapReduce Execution



```
Node1 startup
Received input
Response sent
```

Figure 5. Reducer Function

the processes are over, it produces zero or more pairs in the final.

Reduce(): Multiple reducers can run in parallel and the number of reducers is specified by the user. Output key will contain the key output from the reducer and output value will contain the value that is output for that particular key.

3.3 Aggregator Function

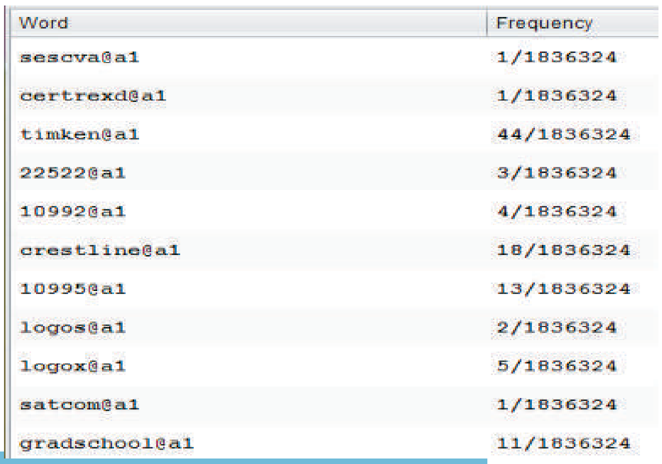
In aggregator function, each aggregator can reduce merged traffic from multiple map tasks. Figure 6 shows the result of the aggregator function.

4. Proposed Algorithm

Optimized Distributed Algorithm:

The steps in the algorithm is as follows.

1. t = 1 and Lagrangian multiplier (v_pj) to uninformed values;
2. repeat step 3 to 5 until t < T
3. distributively solve the sub problem on multiple machines in a parallel.
4. update the values of (v_pj) with the conjugate gradient method, and send the results to all subproblems;
5. t = t + 1



Word	Frequency
sescva@a1	1/1836324
certrexd@a1	1/1836324
timken@a1	44/1836324
22522@a1	3/1836324
10992@a1	4/1836324
crestline@a1	18/1836324
10995@a1	13/1836324
logos@a1	2/1836324
logox@a1	5/1836324
satcom@a1	1/1836324
gradschool@a1	11/1836324

Figure 6. Aggregator Function

4.1 Flow Diagram

As shown in the data flow diagram (Figure 8), the four activities are

- Identify the location,
- Load the data from the mapper,
- Using MapReduce function and placing aggregator to reduce traffic
- Analyze the result.

All these levels or activities are explained briefly in the system architecture (Figure 1).

5. Simulation Results

The simulation is carried out using the MapReduce programming model in Hadoop and Java programming. In this paper, the authors have evaluated the performance of the network traffic cost by using the optimistic distributed algorithm. These properties make

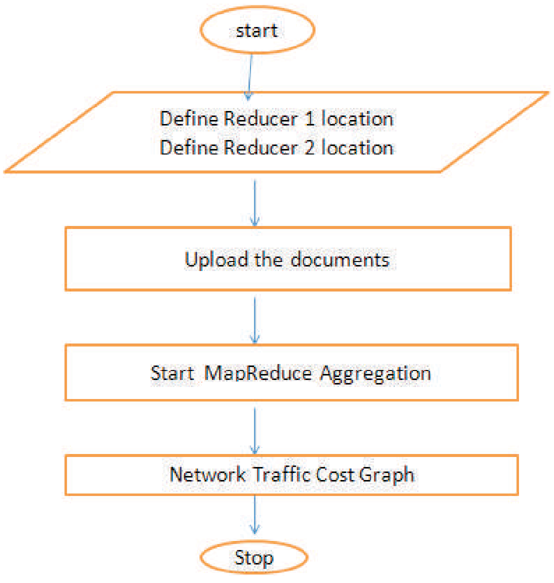

start
Define Reducer 1 location
Define Reducer 2 location
Upload the documents
Start MapReduce Aggregation
Network Traffic Cost Graph
Stop

Figure 7. Flow Diagram

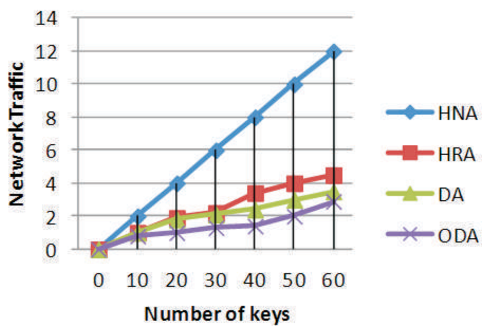


Figure 8. Network Traffic Cost versus Map Tasks

them attractive for many real time systems and applications. The performance of the optimistic distributed algorithm is better than the distributed algorithm. The results of network traffic with map tasks of the optimistic distributed algorithm is mentioned below.

Figure 7 illustrates Network traffic cost versus the number of keys from 1 to 60. In this figure, HNA stands for Hash-based partition with No Aggregation, HRA stands for Hash-based partition with Random Aggregation, DA stands for Distributed Algorithm, and ODA means Optimized Distributed Algorithm.

Conclusion

In this paper, the authors have studied the optimization of intermediate data partition and aggregation for reduced the network traffic cost for large size data applications. They have proposed an optimized distributed algorithm to deal with the large-scale data. Finally, simulation results show that their proposal can significantly reduce network traffic cost. Furthermore, they planned to extend the algorithm to handle the MapReduce job in a Hadoop Multimode cluster.

References

[1]. Ian Foster, Carl Kesselman, and Steven Tuecke, (2001). "The Anatomy of the Grid: Enabling Scalable Virtual Organizations". *International Journal of High Performance Computing Applications*, Vol.15, No.3, pp.200-222.

[2]. Shaik Naseera, T. Vivekanandan, and K.V. Madhu Murthy, (2008). "Data Replication using Experience Based Trust in Data Grid Environment". In *Proceedings of 5th International Conference on Distributed Computing and Internet Technology*, Springer-Verlag, Heidelberg, Vol.5375, pp.39-50.

[3]. MapReduce introduction. Retrieved from". http://www.tutorialspoint.com/map_reduce/map_reduce_introduction. htm

[4]. A. Blanca and S.W. Shin, (n.d.). *Using Network Bandwidth Smartly in MapReduce Scheduling*. Retrieved from http://www.cs.berkeley.edu/~kubitron/courses/cs262a-F13/projects/reports/project7_poster.pdf

[5]. B. Palaniswmy, A. Singh, L. Liu, and B. Jain, (2011). "Purlieus: locality-aware resource allocation for MapReduce in a cloud". In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis.* ACM, p.58.

[6]. P. Costa, A. Donnelly, A. I. Rowstron, and G. O'Shea, (2012). "Camdoop: Exploiting in-network Aggregation for Big Data Applications". In *NSDI*, Vol.12, pp.1-14.

[7]. T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, J. Gerth, J. Talbot, K. Elmeleegy, and R. Sears, (2010). "Online Aggregation and Continuous Query support in MapReduce". In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ACM, pp.1115-1118.

[8]. S. Chen and S. W. Schlosser, (2008). "Map-Reduce Meets Wider Varieties of Applications". *Intel Research Pittsburgh, Tech. Rep. IRP-TR-08-05.*

[9]. F. Ahmad, S. Lee, M. Thottethodi, and T. N. Vijaykumar, (2013). "MapReduce with Communication Overlap". *Journal of Parallel and Distributed Computing*, Vol.73, No.5, pp.608-620.

[10]. Wikispaces (n.d.). *Map Reduce.* Retrieved from https://hadooptutorial.wikispaces.com/MapReduce

ABOUT THE AUTHORS

K. Reddamma is a Research Scholar in the Department of Computer Science and Engineering at Sreenivasa Institute of Technology and Management Studies, Chittoor, India. She obtained her Bachelor's Degree in Computer Science and Engineering from Sri Venkateswara College of Engineering, Tirupati, India. Also she obtained her Master's Degree in Computer Science and Engineering from Sreenivasa Institute of Technology and Management Studies, Chittoor, India. His current research interest is Big Data.

Dr. D. Jagadeesan is currently working as a Professor in the Department of Computer Science and Engineering at Sreenivasa Institute of Technology and Management Studies, Chittoor, India. He obtained his Bachelor's Degree in Computer Science and Engineering from Anna University, Chennai, Tamil Nadu, India. He obtained his Master's degree in Computer Science and Engineering from Dr. M.G.R University at Chennai, Tamil Nadu, India. He obtained his Ph.D in Computer Science and Engineering at Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya (SCSVMV University), Kanchipuram, Tamil Nadu, India. He is Life Member of the ISTE and a Member of the IEEE. He is specialized in Networking, Mobile Ad-hoc Network, Compiler Design and Computation Theory. His current research interests are Route Recovery in MANET, Mobile Routing Protocols, Mobile Heterogeneous Network, Network Security, Big Data, and e-Agriculture.

T. Vivekanandan is currently working as an Associate Professor in the Department of Computer Science and Engineering at SITAMS, Chittoor, India. He has 12 years of teaching experience and 2 years of industrial experience. He has published papers in a reputed National and International Journals and Conferences besides a life member in the Indian Society for Technical Society (ISTE) and Computer Society of India (CSI). His research interest, includes Data Analytics in e-Healthcare, Big Data Analytics, High Performance Computing (HPC), Cloud and Grid Computing.