International Journal of Advance Foundation and Research in Computer (IJAFRC)
Volume 1, Issue 12, December 2014. ISSN 2348 – 4853

Improving MapReduce Performance Via Locality Aware And Asynchronous Processing.

Mohammad Ghoneem, Lalit Kulkarni
Department of Information Technology
Pune University, Pune, India
Mhd89aiu@gmail.com, *lalit.kulkarni@mitcoe.edu.in*

A B S T R A C T

One of the most important keywords for big data and cloud computing in these days is MapReduce. MapReduce can be found as an open source implementation in Hadoop for processing BigData. The logic behind MapReduce is to utilize and explore computing power of all nodes in network. When the master node in the cluster receives a request from the client, ittries to spawn multiple smaller versions of this request known as map function. Each of this map function is assigned to a slave node in the cluster to process the data and get a sub result known as intermediate results. The next operation start going in the reverse direction by collecting the intermediate result and try to merge these results into a final result using reduce function. The current MapReduce implementation in Hadoop can be further improved by enhancing the portion of the code regards the utilization of system resources. In this paper we try to analyze two different techniques regarding network traffic and synchronization between Map and Reduce function. In Hadoop the reduce task does not take into consideration the location and size of each block of intermediate results in order to minimize the traffic in the network during the reduce phase. For that purpose we analyze an algorithm which try to assign reduce task to the node which has the maximum amount of data so that the amount of traffic will be less.The second problem is that the MapReduce in Hadoop does not allow overlapping of map and reduce function which result in inefficient utilization of the resources. To solve such kind of problem we implement an incremental strategy to exploit the parallelism provided by the system.

Index Terms: Hadoop, MapReduce, LARTS, asynchronous processing, incremental strategy

I. INTRODUCTION

In today's world with the evolution in soft technology people start depending on different application in many fields such as commercial and scientific to do their day to day work. These applications start seeking for more and more processing capabilities which can't be provided by a single machine.one of the potential solution to this problem was distributed and parallel processing which can be found as a MapReduce [1] function proposed by Google in 2004.

MapReduce can be found as an open source implementation in Hadoop. Hadoop MapReduce follows master slave architecture [4] in which a user submits a query to master node in the cluster. The master node then splits this query into sub queries known as map functions and assigns these map functions to the slave nodes which contain the block of data to be processed. Each slave node with map function process the data accordingly and return a result known as intermediate result [5]. Here the first phase of MapReduce is completed. The second phase start when master node get a notification from all slave nodes that map function is done by sending heart beat message [4]. Then the master node assigning a reduce function to n number of slave nodes. The slave node then start merging the intermediate results and apply the reduce function to get the final result.

The current implementation of MapReduce in Hadoop can be further enhanced by improvingthe scheduling technique of MapReduce and the level of parallelism between map and reduce function (utilization of resources).in case of network traffic it might become the bottleneck of MapReduce model especially during the shuffling of large size of intermediate results to the node with the reduce function which might degrade the performance. Many schedulers have been developed for scheduling the MapReduce tasks in Hadoop. First scheduler is the default Hadoop scheduler [2] which implement FIFO algorithm. In this scheduler the jobs are executed according to the order of their submission. Another scheduler is the Fair Scheduler [2] which has been developed by Facebook. The aim of this scheduler is to give a fair share to the cluster capacity over time. Yahoo also has developed a scheduler called Capacity Scheduler. This scheduler tries to ensure fair allocation of computation resources when the number of users is large [2]. All of these schedulers take care of map function scheduling by scheduling the task at the closest node which contain the block of data to be processed. Master node schedules the task by giving the priority to in-rack-nodes and if not possible at these nodes it will assign it to out-rack-node. But in case of reduce function Hadoop does not provide any mechanism or algorithm in assigning reduce task to get the minimum traffic in the cluster.

By applying the LARTS and IR strategy to the current implementation of MapReduce in Hadoop we can overcome two problems. First the network traffic in the cluster will be minimized. Second the resources in the cluster will be more efficiently utilized by increasing level of parallelism. As a result the overall MapReduce performance in Hadoop will be improved.

II. HADOOP SCHEDULING TECHNIQUES

A. MapReduce scheduling

In Hadoop master slave architecture the master node must execute a program known as a job Tracker (JT) while the slave nodes must run a program known as task tracker (TT) [4]. These two programs provide the communication between master and slave nodes in the cluster. A heartbeat message is used by the task tracker in the slave node to communicate with the master node (JT). The heartbeat message might contain information about the slave node indicating the state of the task (map or reduce) running at that node.

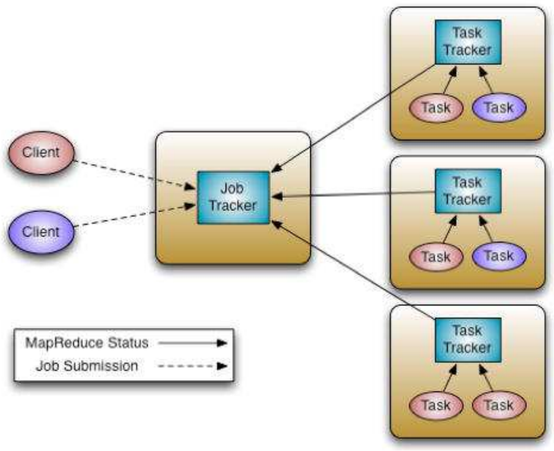

Figure 1: MapReduce Model

The heartbeat message [4] is also used by slave nodes to send to send a request to master node to get a task (map or reduce). Then the job tracker at the master node replies by assigning a task to that task tracker at the salve node. When job tracker receive a request from the slave nodes it try to schedule the map task according to their vicinity of input split [8], but in case of reduce task job tracker assign it to the slave nodes without taking any consideration to size and location of data blocks that need to be transferred in the network.

B. Early shuffle in Hadoop

Early shuffle [4] is a technique used to enhance Hadoop performance by starting the shuffle phase early (after 5% of map tasks are completed). This technique will decrease the execution time by over lapping map and shuffle phase. Starting the shuffle phase early means that reduce tasks have been already scheduled at the nodes which had finish their map tasks. These nodes usually produce a small size of data as a result of executing the map tasks. The other nodes in the cluster which still executing map function will produce a larger amount of data. When the reduce phase starts the big size results need to be shuffled in the network which implies more traffic in the network.

C. Recursive Reduction

One of the characteristics of MapReduce jobs are that there is no synchronization requirement between map and reduce phase [5]. We can benefit from this point by starting the reduce phase early after a certain number of map tasks are committed. In the reduce phase the reduce function will be applied to those results which has been received from a certain number of map functions, and store the reduction of the sub result locally. This procedure is repeated and the sub results accumulated until all the map tasks are committed.

To overcome the above mentioned problems first we implement locality-aware reduce task scheduling algorithm [4]. In this solution we try to provide the algorithm with information about data size and its location in the network. Then the algorithm submits reduce tasks to the nodes which have the largest amount of data to minimize network traffic during shuffling phase. Second enhancement can be done to the current map reduce by increasing level of parallelism between map and reduce functions using incremental reduction strategy [5].
In section III we describe how the LARTS [4] algorithm can minimize the traffic in Hadoop cluster. Then in section IV we describe the Incremental Reduction [5] mechanism and how it utilizes high level of parallelism in Hadoop cluster.

III. LARTS SCHEDULER FOR TRAFFIC MINIMIZATION

In Hadoop environment the size of data transferred in the network depends on where reduce tasks have been scheduled. In the cluster if we have only one mapper and one reducer, the best way to schedule them so that the traffic in the network is minimum is to schedule mapper and reduce at the same node (no traffic). But in real world situation there are many mappers that feed one reducer. In this case we have to schedule the reducer at the node which feed the reducer with maximum amount of data. But if it is not possible to schedule the reducer at the maximum node then the best option is to schedule it at one of the nodes in the maximum rack (maximum rack is the rack which feed the reducer with maximum amount of data).Following such kind of scheduling procedure will result in scheduling delay and low level of resource utilization and parallelism.

The Job Tracker rejects requests from Task Trackers 1 and 3 because they don't satisfiy LARTS's conditions (JT = Job Tracker, TT*j*= Task Tracker *j*, RC*TTj*= rejection counter associated with TT*j*, R*i*= reducer *i*, and P*Ri*:*TTj*= partition P produced at TT*j*and hashed to reducer, R*i*) [4].

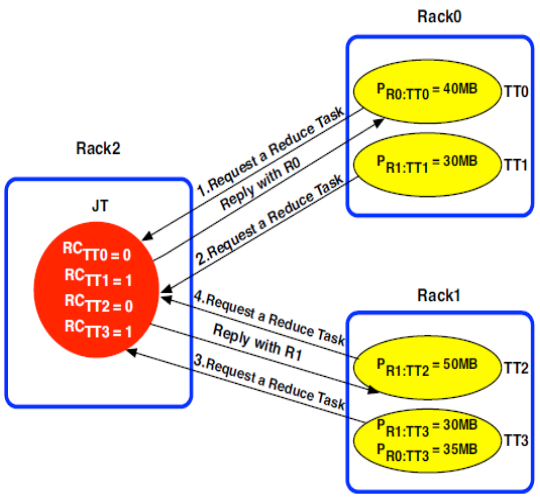

Figure 2: An example of Task Trackers making requests for reduce tasks in LARTS.

The scheduling delay happens when master node wait for maximum node to send a request to get a reducer task, while rejecting the other requests from the other nodes. The low level of resource utilization and parallelism will happen because some of the nodes in the cluster will not be preferred to any of the reduce tasks.

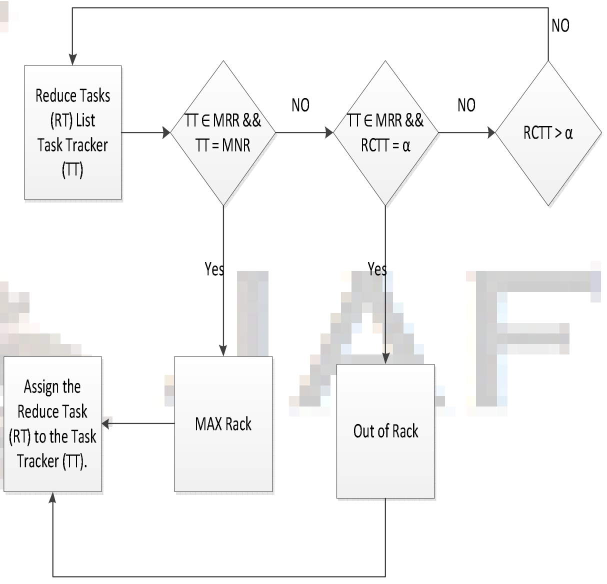
Reduce Tasks (RT) List Task Tracker (TT)
TT ∈ MRR && TT = MNR
TT ∈ MRR && RCTT = α
RCTT > α
NO
NO
NO
Yes
Yes
Assign the Reduce Task (RT) to the Task Tracker (TT).
MAX Rack
Out of Rack

Figure 3: LARTS Work Flow.

To overcome these two problems a rejection counter for each task tracker is introduced by the LARTS algorithm and to control and limit this counter from passing a certain value we introduce a threshold α. [4] When the algorithm runs it try to assigns the reduce task to the maximum node [4] so when a request comes from any other node it rejects the request and increment the rejection counter of that node by1. After that if rejection counter reaches α value the algorithm will go to first stage of relaxation mode. In this mode the master node try to assign reduce task to one of the nodes which is in the maximum rack [4]. After that if rejection counter become greater than threshold α the LARTS algorithm goes to second stage of relaxation algorithm. In this mode the master node assign the reduce task to any of the nodes in the cluster without taking into consideration the above conditions (maximum node, maximum rack) [4].

By implementing the LARTS algorithm the network traffic will be minimized by scheduling the tasks at maximum node or maximum rack which implies less time for data shuffle (as shown in the figure 4). While doing so the scheduling delay and resource utilization will be maintained due to relaxation conditions.

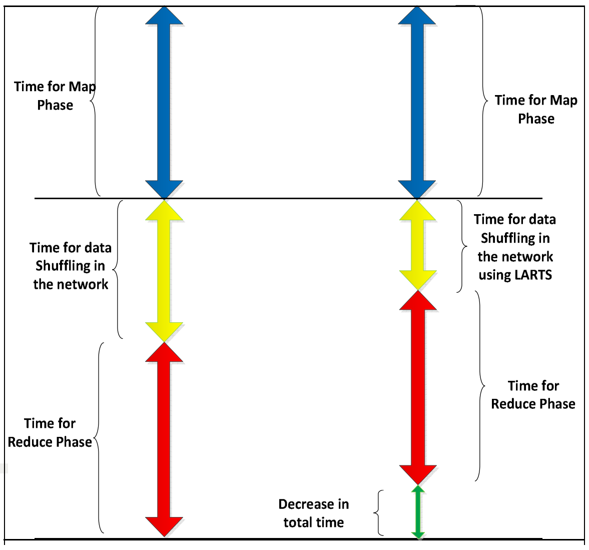

Figure 4: MapReduce Scheduler using LARTS algorithm.

Asynchronous MapReduce

The reduction stage in Hadoop can be divided into three stages. The first stage is the shuffling of intermediate results from all mappers in the network to the reducer node. The second stage is sort and merge in which the intermediate results is sorted and merged according to their key values. Finally in the third stage the reduce function is applied on the merged data to get the final result [5].

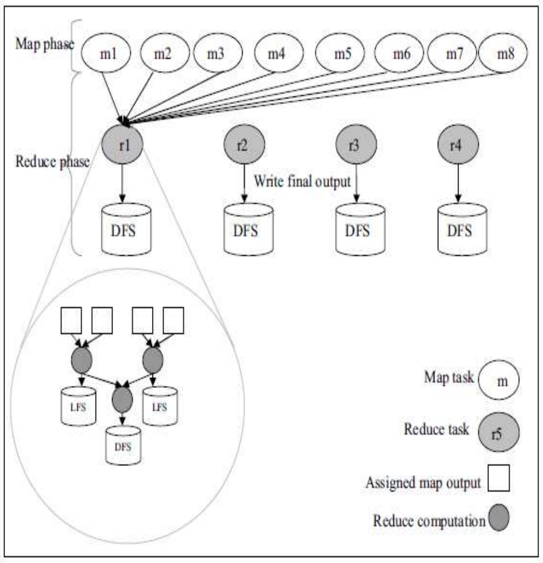

Figure 5: Incremental Reduction mechanism for Reduce phase

Incremental reduction (IR) algorithm improve map reduce performance by over lapping the map and reduce phase [5]. First of all the shuffling stage will start early after a certain number of map tasks commit. When the reducer node receive a certain number of intermediate results the sort and merge stage will start and the merging result stored locally (as shown in figure 5). This procedure repeated until all mapper results are received. After that the reduce function is applied to the locally stored results to get the final result and store it in the distributed file system [5].

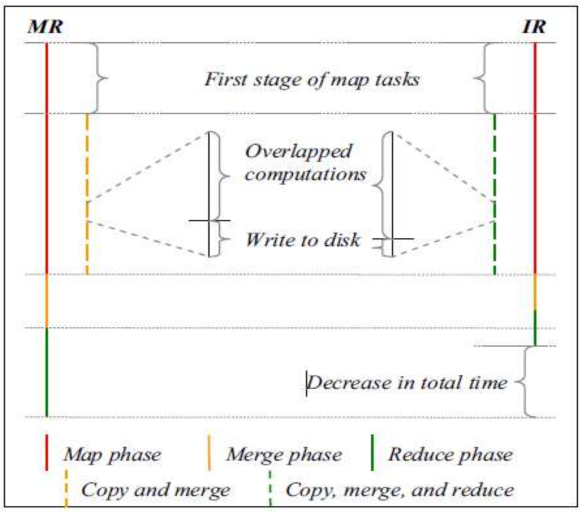

Figure 6: Execution time for MR and IR.

As a result if the over lapping degree is high between the map and reduce phase the final merging and reduce time can be ignored (as shown in figure 6) so that the execution time is significantly decreased.

IV. CONCLUSION

MapReduce is a very powerful model for huge data processing but still further improvement can be done. In this paper two techniques are discussed, which are used to improve the performance of MapReduce model. The first improvement is to make reduce task aware about the size and location of map tasks output in order to minimize network traffic. This can be done by scheduling the reduce task at the node which feed it with maximum amount of data and if not possible the reduce task can be scheduled at the maximum rack. The second improvement is to exploit the level of parallelism in the cluster using incremental reduction mechanism. IR decreases the execution time significantly by over lapping the map and reduce phase. Using these two mechanisms the overall MapReduce performance will be improved.

V. REFERENCES

[1] Jeffrey Dean and Sanjay Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters, ”In Communications of the ACM, Volume 51, Issue 1, pp. 107-113, 2008.J.Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon,1892, pp.68–73.

[2] Thirumala Rao and Dr. L.S.S. Reddy,"Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments".

[3] Shyam Deshmukh and Dr. J. V. Aghav, "Job Classification for MapReduce Scheduler in Heterogeneous Environment ".

[4] Mohammad Hamoud and Majd F,Sakr,"Locality Aware Reduce Task Scheduling for MapReduce ".

[5] Marwa Eltier and Heshanlin ,"Enhancing MapReduce via Asynchronous Data Processing".

[6] B.ThirumalaRao and N.V.Sridevi,"Performance Issues of Heterogeneous Hadoop Cluster in Cloud Computing".

[7] Shamil Humbetov ,"Data Intensive Computing with MapReduce and Hadoop".

[8] S. Ghemawat, H. Gobioff, and S. T. Leung, ”The Google File System,” *SOSP,*2003.

[9] S. Kyong-Ha Lee, and Hyunsik Choi, ”Parallel Data Processing with MapReduce: A Survey”.

[10] JaideepDhok and VasudevaVarma,"Using Pattern Classification for Task Assignment in MapReduce ".

[11] Hadoop. http://hadoop.apache.org/

[12] Hadoop Tutorial. http://developer.yahoo.com/hadoop/tutorial/