Volume 4, Issue VIII, Aug. 2016
ISSN 2320-6802

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY

WINGS TO YOUR THOUGHTS.....

A Review on MapReduce Scheduling Algorithm in Hadoop for Heterogeneous Environment

[1]Amandeep Kaur, [2]Neha Kapoor, [3]Sushil Lekhi

[1,2,3] Rayat Institute of Engg. & IT, Ropar, Punjab, India,
[1]amandeep.banwait@gmail.com,[2]nehak189@gmail.com,[3]lekhi.engg@gmail.com

Abstract: *Big data is a buzzword, or catch-phrase, meaning a massive volume of both structured and unstructured data that is so large it is not easy to process using traditional database and software techniques. Hadoop is one of the popular technologies in the big data landscape for evaluating the data through Hadoop Distributed File System and Map-Reduce. Hadoop schedulers are pluggable parts which assign resources to jobs. MapReduce is a programming standard and an associated implementation for converting and generating large datasets. It allow users to specify a map function which processes a key/value pair to create a set of in-between key/value pairs, and a reduce function that combines all the intermediate values linked with the same intermediate key. This paper presents the overall study about Big Data, Hadoop technology using Hadoop Distributed File System and MapReduce, their architectures, problem identification, problem statement and the recommended technology. Self-adaptive map reduce scheduling algorithm is used in the recommended technology.*

Keywords: *Big Data, Hadoop, MapReduce, Map Tasks, Reduce Tasks, HDFS, SAMR, Progress Score.*

1. INTRODUCTION

Big data has established an era of tera where huge volume of data is being gathered at fascinating rates. Due to hike in storage capacities, processing power and availability of data, the size of global data is increasing in zeta-bytes. Hadoop is one of the popular technologies in the big data landscape for evaluating the data through Hadoop Distributed File System and Map-Reduce. Job scheduling is very important activity for proper management of cluster resources. Hadoop schedulers are pluggable parts which assign resources to jobs. In various types of schedulers, popular are the FIFO, Fair and Capacity schedulers.

MapReduce is a programming standard and an associated implementation for converting and generating large datasets. It allow users to specify a map function which processes a key/value pair to create a set of intermediate key/value pairs, and a reduce function that combines all the intermediate values linked with the same intermediate key. Map Reduce has been started by Google, in association with GFS and Big Table comprising backbone of Google's Cloud Computing platform. Map Reduce has achieved great success in various applications counting from horizontal and vertical search engines to GPU to multiprocessors. MapReduce has been considered as one of the key empowering methodologies for taking care of ceaselessly increasing requests on figuring assets forced by Big Datasets yet at the same time many issues arrive with MapReduce keeping in mind the end target to handle a much more extensive cluster of employments, combination into Hadoop's native file system. The objective behind this is the high versatility of the MapReduce worldview which considers hugely parallel and circulated execution over an expansive number of figuring hubs.[4]

2. HADOOP: SOLUTION FOR BIG DATA PROCESSING

Hadoop was mainly designed for huge batch jobs which did not need human interventions. With the rise in the number of users, and as organizations started using more data and computations in their Hadoop clusters, workload increased and therefore the method of sharing of clusters was developed for higher resource utilization. During earlier scheduling strategies, each job would use the whole cluster resulting job starvation. Many tiny jobs like sampling, adhoc queries and periodic reports consisting of many jobs which required small time to run were starved. Shared cluster having a multi-user solution for sharing the resources among all the users.[1]

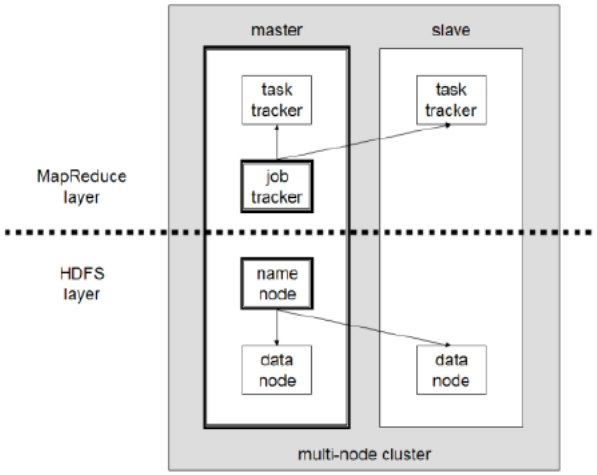

Figure 1: Hadoop Architecture

2.1 Hadoop Distributed File System Architecture

Hadoop includes a fault-tolerant storage system called the Hadoop Distributed File System, or HDFS. HDFS can store huge amounts of information, scale up iteratively and survive the failure of significant parts of the storage infrastructure without any data loss. Hadoop creates *group* of machines and control work among them. Clusters can be created with inexpensive computers. If one fails, Hadoop continues to use the cluster without losing data or interfering work, by shifting work to the left machines in the cluster. HDFS handles storage

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS…..

on the cluster by breaking incoming files into parts, called "blocks," and storing each of the blocks redundantly across the pool of servers. In general case, HDFS stores three full copies of each file by copying each part to three different servers.[3][7]

2.2 MapReduce Architecture

MapReduce is a programming model enabling many of nodes to handle huge data by cooperation. In traditional MapReduce scheduling algorithm, a MapReduce application needs to be run on the MapReduce system is called a "job". A job can be divided into a series of "Map tasks" (MT) and "Reduce tasks" (RT). The tasks which execute map function are called "Map tasks", and which execute reduce function are called "Reduce tasks". In a cluster which runs MapReduce, nodes were classified into "NameNode" and "DataNode". The processing pillar in the Hadoop ecosystem is the MapReduce framework. The framework allows the specification of an operation to be applied to a huge data set, divide the problem and data, and run it in parallel. From an analyst's point of view, this can occur on multiple dimensions. For example, a very large dataset can be reduced into a smaller subset where analytics can be applied. In a traditional data warehousing scenario, this might entail applying an ETL operation on the data to produce something usable by the analyst. In Hadoop, these kinds of operations are written as MapReduce jobs in Java.[6] There are a number of higher level languages like Hive and Pig that make writing these programs easier. The outputs of these jobs can be written back to either HDFS or placed in a traditional data warehouse. There are two functions in MapReduce as follows:

Map – the function takes key/value pairs as input and generates an intermediate set of key/value pairs

Reduce – the function which merges all the intermediate values associated with the same intermediate key

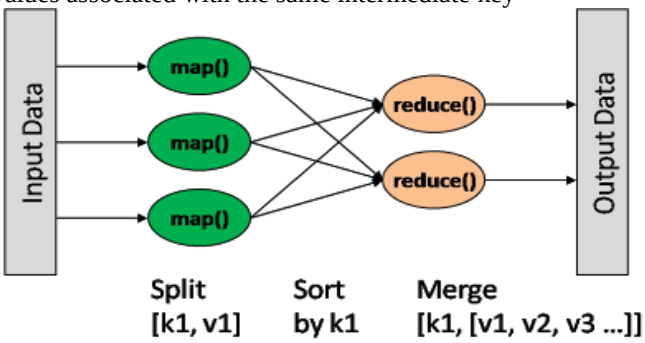

Figure 2: Map Reduce Architecture

3. PROBLEM FORMULATION

In previous work Self-Adaptive MapReduce scheduling algorithm (SAMR) has been implemented. SAMR: a Self-Adaptive MapReduce scheduling algorithm [2], which evaluates progress of tasks dynamically and adapts to the constantly changing environment automatically. SAMR is developed with a same idea to LATE MapReduce scheduling algorithm. However, SAMR gets better PS values of all the tasks by using historical information. By using accurate PSs, SAMR reveals real slow tasks and decreases execution time compared with Hadoop and LATE.

3.1 SAMR Algorithm:

- Procedure SAMR.
- Input: Key/Value pairs.
- Output: Statistical results.
- Reading historical information and tuning parameters using it.
- Finding slow tasks.
- Finding slow Task Trackers.
- Launching backup tasks.
- Collecting results and updating historical information.
- End procedure.

The experimental results have shown the efficiency of self adaptive MapReduce scheduling algorithm. The algorithm reduces the execution time of MapReduce jobs, especially in heterogeneous environments. The algorithm chooses slow tasks and launch backup tasks accordingly while categorising nodes correctly, and saving huge system resources. The most important contributions of the SAMR algorithm are: SAMR uses previous information recorded on each node to tune weight of every stage dynamically. SAMR takes the two stages features of map tasks into consideration for the first time. SAMR categorise slow nodes into map slow nodes and reduce slow nodes further.[5]

4. PROBLEM STATEMENT

The most popular implementation of MapReduce, Hadoop, suffers from an issue that cannot distinguish tasks which need backup tasks on fast nodes properly. Hadoop supervise the progress of tasks using "Progress Score (PS)" (range from 0 to 1). The average progress scores PSavg. The Progress Score of the ith task is PS[i]. Consider the number of tasks which are being executed is T, the number of key/value pairs required to be processed in a task is N, the number of key/value pairs have been processed in the task is M, and the task has finished K stages (only for reduce task. There are three stages in a reduce task: copy data phase, sort phase and reduce phase).

LATE MapReduce scheduling algorithm always starts backup tasks for those tasks which have more time left than other tasks. Although LATE uses an effective strategy to start backup tasks, it mainly launches backup tasks for inappropriate tasks. This is because LATE cannot get TTE for all the running tasks properly. Another drawback of LATE MapReduce scheduling algorithm is that it does not categories map slow nodes and reduce slow nodes. One node may executes MT quickly, but executes RT slower than most of other nodes. LATE just assume one node either fast node or slow node, does not classify slow nodes further.[10]

SAMR is developed with a same idea to LATE MapReduce scheduling algorithm. However, SAMR gets proper PSs of all the tasks by using historical information. By using accurate PSs, SAMR reveals real slow tasks and decreases more execute time compared with Hadoop and LATE.

The SAMR algorithm could be further enhanced in term of several aspects. First, this algorithm will consider on how to

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

account for data locality when launching backup tasks, because data locality may remarkably accelerate the data load and store. Second, SAMR is considering a mechanism to incorporate that tune the parameters should be added. Third, SAMR will be evaluated on various platforms by first evaluated on rented Cloud Computing platform.[9]

5. PROPOSED METHODOLOGY

Step:-1 Read historical information and tuning parameters using historical information.

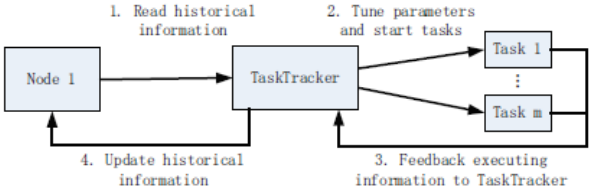

Figure 3: Historical information use and update

Step:-2 Find slow task.
Step:-3 Find slow Task Trackers.
Step:-4 Launch backup task.[8]

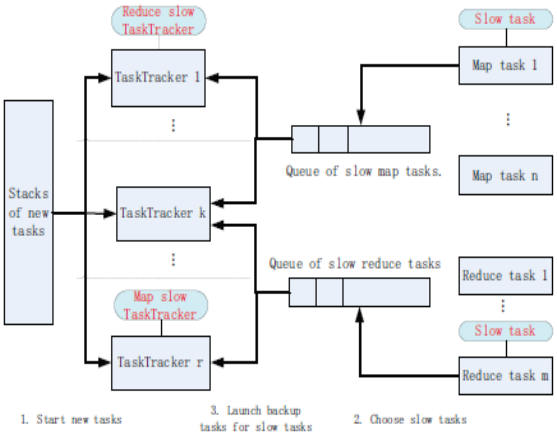

Figure 4: SAMR Overview

6. CONCLUSION

This paper gives us a brief introduction of big data, Hadoop along with Hadoop distributed file system architecture and MapReduce architecture. In this paper, we have concluded that the SAMR algorithm reduces the execution time of MapReduce jobs, especially in different environments. The algorithm chooses slow tasks and launch backup tasks accordingly while categorizing nodes correctly, and saving huge system resources. The proposed algorithm will focus on how to account for data locality when initiating backup tasks, because data locality may remarkably accelerate the data load and store.

REFERENCES

[1] Harshawardhan S. Bhosale and Devendra P. Gadekar, "A Review paper on Big Data and Hadoop," in International Journal of Scientific and Research Publications, Volume 4, Issue 10, October 2014.

[2] Quan Chen and Song Guo, "SAMR: A Self-adaptive MapReduce Scheduling Algorithm in Heterogeneous Environment", IEEE International Conference on Computer and Information Technology, 2010

[3] Jobby P Jacob and Anirban Basu, "Performance Analysis of Hadoop Map Reduce on Eucalyptus Private Cloud", International Journal of Computer Applications (0975 – 8887), Volume 79 – No 17, October 2013.

[4] Anjana Gosain and Nikita Chugh, "New Design Principles for Effective Knowledge Discovery from Big Data", International Journal of Computer Applications (0975 – 8887), Volume 96– No.17, June 2014

[5] Namrata Singh and Sanjay Agrawal, "A REVIEW OF RESEARCH ON MAPREDUCE SCHEDULING ALGORITHMS IN HADOOP", International Conference on Computing, Communication and Automation (ICCCA2015)

[6] C.J. Kavithapriya, "An Imminent Approach for Genome Sequence and Analysis using Map Reduce", International Journal of Computer Applications (0975 – 8887) Volume 128 – No.7, October 2015

[7] Sushma Lakshkar, Geet Kalani and Vinod Todwal, "A Catholic Research on Big Data and Hadoop Environment", International Journal of Computer Applications (0975–8887), Volume 130–No.11, Nov. 2015.

[8] Jyotsna Talreja Wassan, "Modeling Stack Framework for Accessing Electronic Health Records with Big Data Needs", International Journal of Computer Applications (0975 – 8887) Volume 106–No.1, Nov. 2014.

[9] Vikram Yadav, Pooja Malik and G. Sahoo, "Energy Efficient Virtual Machine Optimization", International Journal of Computer Applications (0975 – 8887) Volume 106 – No.7, November 2014.

[10] Hamoud Alshammari and Hassan Bajwa. "Improving Current Hadoop MapReduce Workflow and Performance", International Journal of Computer Applications (0975–8887), Volume 116–No. 15, April 2015.

[11] Vishal Dubey, Saumya Gupta and Sapeksh Garg, "Performing Big Data over Cloud on a Test-Bed", International Journal of Computer Applications (0975–8887), Volume 120–No.10, June 2015.