International Journal of Computer Applications (0975 – 8887)
Volume 121 – No.12, July 2015

Enhancing Dynamic Capacity Scheduler for Data Intensive Jobs

Sukhmani Goraya
Department of Computer Science and Engineering
CT Institute of Engineering, Management and Technology, Shahpur, Jalandhar, India.

Vikas Khullar
Department of Computer Science and ngineering
CT Institute of Engineering, Management and Technology, Shahpur, Jalandhar, India.

ABSTRACT

Management of Big Data is a Challenging issue. The MapReduce environment is the widely used key solution for data intensive jobs. We will analyze map reduce pipelining and along with processing of Map phase and Reduce phase. Core schedulers FIFO, Fair and Capacity Schedulers have been discussed. The Scheduler assigns MapReduce task to the resources and there is a challenge to the scheduler to schedule the task in a way that it is scalable. Existing work shows the performance of the Hadoop depends upon input data and configuration of the cluster. In this paper, we have analyzed the execution time for data intensive jobs with increasing volume of the data set. We have also compared the execution time of the task with existing scheduler and our proposed method for the scheduler.

Keywords

Hadoop, MapReduce, Capacity Scheduler, Fair Scheduler, FIFO Scheduler, HDFS.

1. INTRODUCTION

Hadoop can be described as architecture for large scale computation and data processing on a network. High scalability and flexibility are the major advantages which allow users for large amount of data processing benefiting a number of fields such as machine learning, security and bioinformatics. IT companies is creating large amount of data which is in Terabytes (TB) and Petabyte (PB). As a platform of computational and data storage , Hadoop can handle many different types of data including file format such as audio; video; text; e-mail records; images etc. MapReduce is currently the most famous framework for processing large sets of data in parallel. Hadoop clusters run on inexpensive hardware, so that the projects can scale-out without spending more cost.

Hadoop MapReduce is the phenomenon of processing large data sets of the computer cluster. There are two phases, Map phase and Reduce phase.

1.1 MapReduce

In Map phase, the problem gets divided into the sub problem and if required it further divides that sub problem. In Reduce phase, the answer to all the sub-problems are calculated and collected. There is also Job Tracker and Task Tracker. Client gives request to the Job Tracker. Further Job Tracker assigns the job to the Task Tracker and Task Tracker does that work by using Map phase and Reduce phase. Further, we have is the benchmark example of WordCount.

Figure 1 shows there are two input files that are passed through the Map phase. The Map phase separates the file by using the function such as extraction, filtering and sorting. In case of the word count example, the map function running on the data node reads the input file and will split the file into blocks. While splitting is done, the map phase assigns every word with a value. For each word encountered, the key would be the word and the value would the number

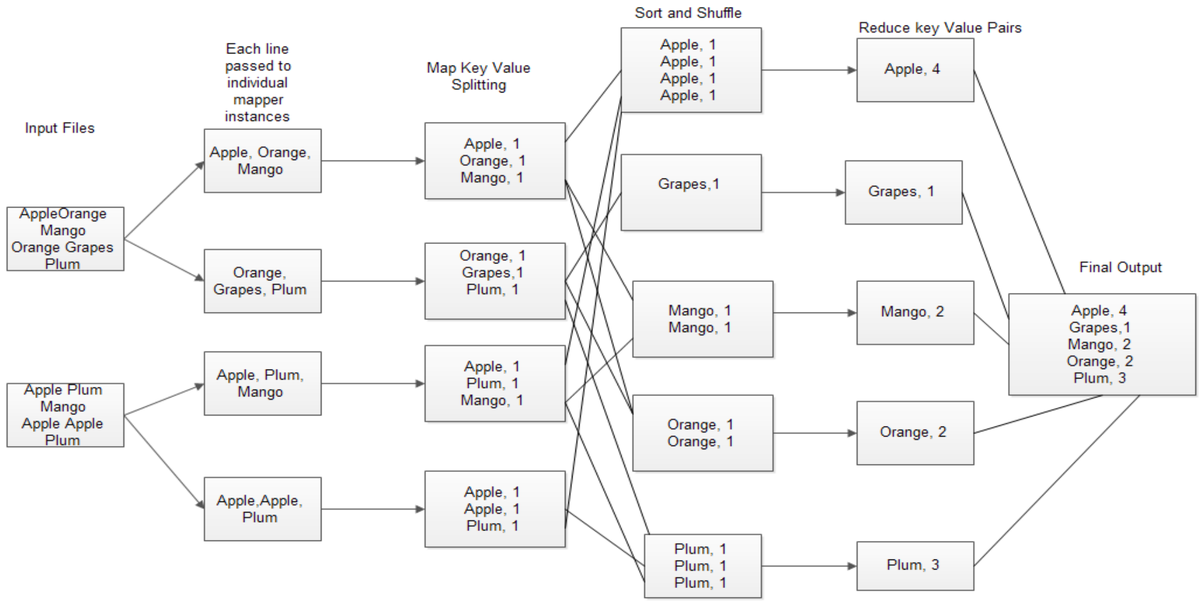

Input Files
AppleOrange Mango Orange Grapes Plum
Apple Plum Mango Apple Apple Plum
Each line passed to individual mapper instances
Apple, Orange, Mango
Orange, Grapes, Plum
Apple, Plum, Mango
Apple,Apple, Plum
Map Key Value Splitting
Apple, 1 Orange, 1 Mango, 1
Orange, 1 Grapes,1 Plum, 1
Apple, 1 Plum, 1 Mango, 1
Apple, 1 Apple, 1 Plum, 1
Sort and Shuffle
Apple, 1 Apple, 1 Apple, 1 Apple, 1
Grapes,1
Mango, 1 Mango, 1
Orange, 1 Orange, 1
Plum, 1 Plum, 1 Plum, 1
Reduce key Value Pairs
Apple, 4
Grapes, 1
Mango, 2
Orange, 2
Plum, 3
Final Output
Apple, 4 Grapes,1 Mango, 2 Orange, 2 Plum, 3

Fig. 1: Word Count Example showing Map Phase and Reduce Phase

of times that word had occurred. Here the map function will only be aware of the currently parsed word. It does not have any memory for prior processing-number of occurrences of that word is 1. Further we can see how the Reduce phase count words having the same key value are moved towards the reduced function. The input to the reduced function is the sorted data from each map function, further reduce function calculates the number of occurrences and reduces it to the final output. The output of reduce function is given to Hadoop Distributed File System.

1.2 HDFS

In a Hadoop cluster, HDFS (Hadoop Distributed File System) is used in which blocks of files are equally distributed among nodes to carry out the task. Hadoop File System uses the block size of 64 MB. Hadoop has the ability for pluggable schedulers that Assign resources to the job.

In case of Hadoop version 1 default scheduling algorithm uses FIFO scheduler, in which JobTracker pulls jobs from a work queue, oldest job first. Fair scheduler, each job gets equal share of available resources so that a single node is not overloaded. Capacity scheduler works for the same principles of Fair scheduler with the slight difference. But in case of Hadoop version 2, default scheduling algorithm is Capacity Scheduler. Each queue is assigned a guaranteed capacity and each queue properties can change at the run time. Task Scheduling technology [1], one of the key technologies of Hadoop platform, mainly controls the order of task running and the allocation of computing resources, which is directly related with overall performance of the Hadoop platform and system resource utilization.

In this paper, we will increase the performance of system by decreasing the execution time which will further reduce the cost of execution. We will mainly focus on Capacity scheduler. Further enhancement of the scheduler will be focused upon. The rest of the paper is organized as follows: Section II presents the background for this research. Section III our methodology and implementation is presented, Section IV exhibits the experimental results, Section V, presents the conclusion and the future work.

2. RELATED RESEAECH WORK

Hadoop is developed by large number of contributors to process large data parallel. It is an Open source implementation. Hadoop consists of the two typical components: Hadoop Distributed File System (HDFS) mimicking Google File System (GFS) [2] and Hadoop MapReduce. MapReduce consists of two functions Map function and Reduce function. Map function divides the problem and solves it. Reduce function collects the output from Map function and reduces it into final output. Hadoop is provided with three job schedulers: Job Queue Task Scheduler, Fair Scheduler and Capacity Task Scheduler. The user has an option of selecting among the users. Job Queue Task Scheduler, which is the base of other job schedulers and the default job scheduler based on First in First out FIFO queue. Tasks are assigned to nodes which maintain their input split with first priority (Data-Local), or other nodes nearby such nodes which maintain their input split with second priority (Rack-Local). Both of Fair Scheduler and Capacity Task Scheduler are job scheduler’s deal with multiple-users [3]. Chen He Ying Lu David Swanson et.al develops a new MapReduce scheduling technique to enhance map task’s data locality. He has integrated this technique into Hadoop default FIFO scheduler and Hadoop fair scheduler. To evaluate his technique, he compares not only MapReduce scheduling algorithms with and without his technique but also with an existing data locality enhancement technique (i.e., the delay algorithm developed by Facebook). Experimental results show that his technique often leads to the highest data locality rate and the lowest response time for map tasks. Furthermore, unlike the delay algorithm, it does not require an intricate parameter tuning process [4].

There are three known schedulers FIFO, Fair Scheduler and Capacity Scheduler.

2.1 FIFO

In the earliest Hadoop MapReduce computing architecture, the essential job sort is massive batch jobs that a single user submits the job, thus Hadoop use inventory accounting (First in 1st out) rule in early planning algorithm [5]. Initially Hadoop used this scheduler in which there was single queue and jobs were executed sequentially.

2.2 Fair Scheduler

In case of Fair scheduler the resources are fairly allocated between jobs. It has pool where jobs are kept and each pool has equal share of resources. Fair scheduling could be a technique of assignment. Resources to jobs such every job gets, on average, an equal share of resources over time [6].

2.3 Capacity Scheduler

Capacity Scheduler [7] originally developed at Yahoo addresses a usage scenario where the number of users is large, and there is a need to ensure a fair allocation of computation resources amongst users. Capacity scheduler has similar functions as that of Fair scheduler. In Capacity scheduler, other than job pool multiple job queues are created and each queue has configurable number of Map and Reduce slots. Each Queue uses FIFO scheduling with priority. If the queue is heavily loaded, it finds unallocated resources for allocation. Capacity Scheduler supports hierarchy of queues and resources are shared among the sub queues and each user has the limit of some percentage to use the resources. If a queue has serious load, it seeks unallocated resources, then makes redundant resources allotted equally to every job [8]. It also allows priority primarily based programming of jobs in associate degree organization queue [9]. When the application is submitted to the queue, the queue will guarantee the capacity allocated to it.

Queues are monitored and are assigned more free resources beyond its capacity if needed. The foremost advanced among III schedulers is a vital drawback in capability algorithm [10]. If needed queues are monitored and are assigned more free resources beyond its capacity. Creation of the queues is not done automatically, for this the user needs to know about the system information.

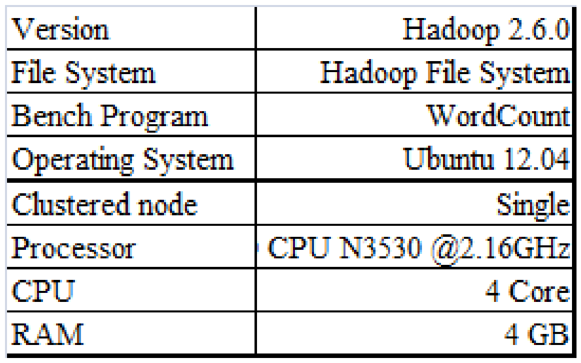
3. IMPLEMENTATION AND METHODOLOGY

The performance of MapReduce basically depends on cluster configuration and input data. We have studied Capacity Scheduler which shares computing resources among the queues. We have performed various experiments on MapReduce application taking benchmark example of word count. The objective of this work is to optimize the task of the scheduler and to get optimal performance from the cluster running MapReduce application.

3.1 Approach

With the advancement Hadoop from version 1 to version 2, there was increase in performance in terms of execution time and capacity of the tasks performed. In Hadoop version 1 fair scheduler was used and in latest version of Hadoop approach of capacity scheduler is used. Although capacity scheduler has enhanced the performance, the limitation was that its limitation is works on the approach of fair scheduler which can be improved further. There is a scope of further increasing the performance of capacity scheduler by enhancing the concept of pipelining. Once the tasks are performed in a synchronous way and performance of capacity scheduler will increase further in terms of both execution time as well as capacity.

TABLE 1: **Performance Environment**



Version	Hadoop 2.6.0
File System	Hadoop File System
Bench Program	WordCount
Operating System	Ubuntu 12.04
Clustered node	Single
Processor	CPU N3530 @2.16GHz
CPU	4 Core
RAM	4 GB

For experiment we have a single clustered node and Hadoop 2.6.0 on operating system Ubuntu 12.04. We are using Bench mark example of WordCount.

3.2 Methodology

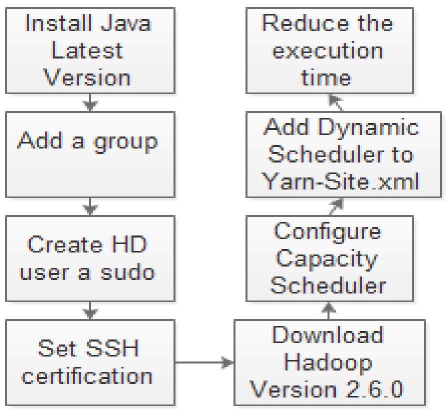

Install Java Latest Version
Add a group
Create HD user a sudo
Set SSH certification
Download Hadoop Version 2.6.0
Configure Capacity Scheduler
Add Dynamic Scheduler to Yarn-Site.xml
Reduce the execution time

Fig.2: Methodology

The methodology used for setting up the system has been shown in the figure 2. Firstly we need to install Java. Then we need to create a group for Hadoop users. SSH certification is used for security purpose in Hadoop. Secondly we need to install Hadoop. After configuring Hadoop, we need to add Dynamic Scheduler to yarn-site.xml.

Once the system is set the Resource Manager gets started, everything is happening in the form of events. Capacity scheduler registers itself with the events and acts on those events. When the node is added, ResourceTrackService registers with the node manager. When application or job is added, it will be submitted to the queue. We have added capacity scheduler to the file yarn-site.xml for processing Hadoop files. Once the scheduler starts, the method run on Capacity scheduler gets started. The Component container of the Resource Manager takes the responsibility of all the resources like disk, CPU, memory, etc. Then the job is given to the existing queue and it is solved in the hierarchical way. Now, the queues are being created by the method addNewCSQueue. Comparison is done between the queues and capacity allocated to the Queues is being calculated by the method CSComparator. The synchronization is between the queues is done by using the method SynchroniseCSQueue. While synchronizing, the queues get information about the resources, they share resources accordingly.

4. PERFORMANCE EVALUATION

The performance evaluation results with the single clustered node have been evaluated and are shown in the Fig. 3. Our graph compares the default Hadoop and Hadoop implemented by the proposed method with the execution time shown in the bar chart. The benchmark example WordCount has been executed with different data sets. The sizes of the data sets are 12 Mb, 36Mb, 300 Mb, 500 Mb, 1 GB, 1.5 GB and 2 GB.

Table 2: Experimental Results

File Size (MB)	Default Hadoop (Execution Time)	Proposed Method (Execution Time)
12.2	0:00:40	0:00:39
36.5	0:00:52	0:00:53
146	0:01:50	0:01:51
300	0:02:08	0:01:52
500	0:02:29	0:02:06
938	0:04:41	0:04:06
1500	0:06:06	0:05:02
2000	0:08:50	0:07:16

Default Hadoop defines the systems default configuration and Proposed Method defines the systems that we have created.
The above table shows the execution time of Default Hadoop and Proposed Method on Hadoop. The proposed method execution is better than the Default settings of Hadoop.

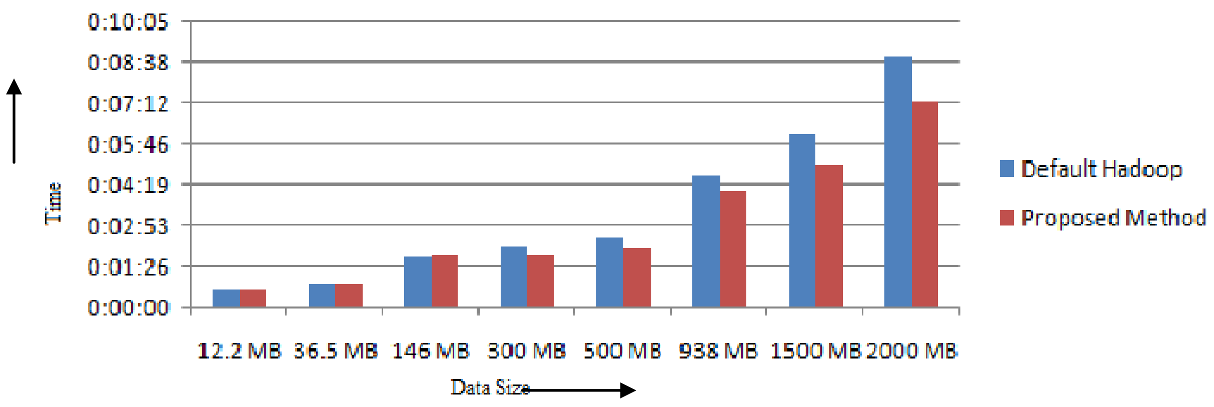


Fig3: Execution Time of Default Hadoop and our Proposed Method

The above graph shows the comparison between the execution time of Default setting of Hadoop and the proposed method of Hadoop. It is observed that when the size of the data is small, there is no such difference between the execution time. But as the size of data increases, the execution time of the proposed method decreases.

5. CONCLUSION

Core Schedulers of Hadoop FIFO, Fair and Capacity have been discussed. Our proposed method has been implemented on 2.6.0 and evaluation has been done by executing the jobs with the benchmark example WordCount. In this paper, we have analyzed that the execution time for data intensive jobs are compared with the existing capacity scheduler and our proposed method for the capacity scheduler. Our proposed method has improved the execution time.

6. ACKNOWLEGEMENT

The authors gratefully acknowledge the support of DRC Team at CT Institutes of Engineering Management & Technology.

7. REFERENCES

[1] Jilan Chen, Dan Wang and Wenbing Zhao," A Task Scheduling Algorithm for Hadoop Platform" in Journal of Computers , vol. 8, no. 4, April 2013.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *19th ACM Symposium on Operating Systems Principles*, Lake George, NY, Oct. 2003.

[3] Shiori KURAZUMI *, Tomoaki TSUMURA *, Shoichi SAITO * and Hiroshi MATSUO *, "Dynamic processing slots scheduling for I/O intensive jobs of Hadoop MapReduce" in 2012 Third International Conference on Networking and Computing.

[4] Harshawardhan S. Bhosale1 , Prof. Devendra P. Gadekar2 "A paper on Big Data and Hadoop" in International Journal of Scientific and Research Publications, Volume 4, Issue 10, October 2014.

[5] M. Isard, M. Budiu, Y. Yu, "*Distributed Data-Parallel Programs from Sequential Building Blocks*," Proceedings of the 2nd ACM SIGOPS European Conference on Computer Systems, ACM, 59-72. 2007.

[6] Hadoop's Fair Scheduler. https://hadoop.apache.org/docs/r1.2.1/fair_sche Duler. [As accessed on 9 Feb. 2015].

[7] Y. Chen, S. Alspaugh, and R.H. Katz, ""Interactive Analytical Processing in Big Data Systems: A Cross-Industry Study of Mapreduce Workloads,"" Proc. VLDB Endowment, vol. 5, no. 12, Aug. 2012.

[8] Umesh V. Nikam, Anup W. Burange, Abhishek A. Gulhane, "*Big Data and HADOOP: A Big Game Changer*", International Journal of Advance Research in Computer Science and Management Studies, Volume 1, Issue 7, ISSN: 2321-7782, DEC 2013.

[9] N. Tiwari, "Scheduling and Energy Efficiency Improvement Techniques for Hadoop Mapreduce: State of Art and Directions for Future Research (Doctoral dissertation, Indian Architectures, algorithms and programming." IEEE; 2011. p. 213–17.

[10] MapReduce NextGen aka YARN aka MRv2 http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn- site/CapacityScheduler.html