© 2015 IJEDR | Volume 3, Issue 2 | ISSN: 2321-9939

Scheduling Techniques for Workload Distribution in YARN Containers

[1]Rajneesh Kumar, [2]Dr.S.Govindarajan
[1]Student, [2]Professor
Department of Computer Application, Faculty of Engineering and Technology, SRM University, Chennai , INDIA

***Abstract* - Big Data, the analysis of large quantities of data to gain new insight has become a ubiquitous phrase in recent years. Day by day the data is growing at a staggering rate. One of the efficient technologies that deal with the Big Data is Hadoop, which will be discussed in this paper. Hadoop, for processing large data volume jobs uses MapReduce programming model. Hadoop makes use of different schedulers for executing the jobs in parallel. The default scheduler is FIFO (First In First Out) Scheduler. Other schedulers with priority, pre-emption and non-pre-emption options have also been developed. As the time has passed the MapReduce has reached few of its limitations. So in order to overcome the limitations of MapReduce, the next generation of MapReduce has been developed called as YARN (Yet Another Resource Negotiator). So, this paper provides a survey on Hadoop, few scheduling methods it uses and a brief introduction to YARN.**

***Keywords*—Hadoop, HDFS, MapReduce, Schedulers, YARN**

I. Introduction

In present scenario with the internet of tilings a lot of data is generated and is analyzed mainly for business intelligence. There are various sources of Big Data like social networking sites, sensors, transactional data from enterprise applications/databases, mobile devices, machine- generated data, huge amount of data generated from high definition videos and many more sources. Some of the sources of this data have vital value that is helpful for businesses to develop. So the question arises how such a gigantic amount of data can be dealt out? Further, there is no stopping of this data generation. There is a great demand for improving the Big Data management techniques. The processing of this huge can be best done using distributed computing and parallel processing mechanisms. Hadoop [1] is a distributed computing platform written in Java which incorporates features similar to those of the Google File System and MapReduce programming paradigm. Hadoop framework relieves the developers from parallelization issues while allowing them to focus on their computation problem and these parallelization issues are handled inherently by the framework.

In section II we discuss in more detail about Hadoop's two important components HDFS and MapReduce. In section III we discuss Hadoop applications. Section IV discusses about some basic types of schedulers used in Hadoop and scheduler improvements. Further section V talks about technical aspects of Hadoop. Section VI focuses on next generation MapReduce paradigm YARN. Finally section VII concludes the paper after which references follow.

II. Hadoop

Hadoop is a framework designed to work with huge amount of data sets which is much larger in magnitude than the normal systems can handle. Hadoop distributes this data across a set of machines. The real power of Hadoop comes from the fact its competence to scalable to hundreds or thousands of computers each containing several processor cores. Many big enterprises believe that within a few years more than half of the world's data will be stored in Hadoop [2], Furthermore, Hadoop combined with Virtual Machine gives more feasible outcomes. Hadoop mainly consists of i) Hadoop Distributed File System (HDFS): a distributed file system to achieve storage and fault tolerance and ii) Hadoop MapReduce a powerful parallel programming model which processes vast quantity of data via distributed computing across the clusters.

A. HDFS- Hadoop Distributed File System

Hadoop Distributed File System [3] [4] is an open- source file system that lias been designed specifically to handle large files that traditional file system cannot handle. The large amount of data is split, replicated and scattered on multiple machines. The replication of data facilitates rapid computation and reliability. That is why HDFS can also be called as self-healing distributed file system meaning that, if a particular copy of the data gets corrupt or more specifically to say if the DataNode on which the data was residing fails then replicated copy can be used. This ensures that the on-going work continues without any disruption.

HDFS lias master and slave architecture. The architecture of HDFS is shown above in Figure 1. In figure alphabets A, B, C represents data block and D fallowed by a number represents a numbered DataNode. HDFS provides distributed and highly fault tolerant ecosystem. One Single NameNode along with multiple DataNodes is present in a typical HDFS cluster. The NameNode, a master server handles the responsibility of managing the namespace of filesystem and governs the access by clients to files. The namespace records the creation, deletion and modification of files by users. NameNode maps data blocks to DataNodes and manages file system operations like opening, closing and renaming of files and directories. It is all upon the directions of NameNode, the DataNodes performs operations on blocks of data such as creation, deletion and replication. The block size is of

64MB and is replicated into 3 copies. The second copy is stored on the local rack itself while the third on remote rack. A rack is nothing but just a collection of data nodes.

B. Hadoop MapReduce

MapReduce [5] [6] is an important technology which was proposed by Google. MapReduce is a simplified programming model and is a major component of Hadoop for parallel processing of vast amount of data. It relieves programmers from the burden of parallelization issues while allowing them to freely concentrate on application development. The diagram of Hadoop MapReduce is shown in Figure 2 below. Two important data processing functions contained in MapReduce programming are Map and Reduce.

The original data will be given as input to the Map phase which performs processing as per the programming done by the programmers to generate intermediate results. Parallel Map tasks will run at a time. Firstly, the input data is split into fixed sized blocks on which parallel Map tasks are run. The output of the Map procedure is a collection of key/value pairs which is still an intermediate output. These pairs undergo a shuffling phase across reduce tasks. Only one key is accepted by each reduce task and based on this key the processing will be done. Finally the output will be in the form of key/value pairs.

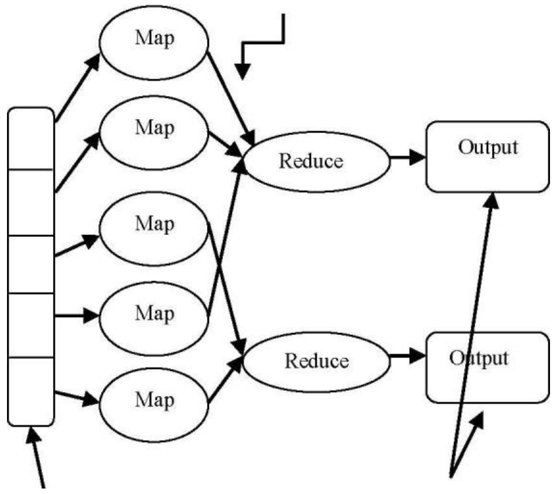

Figure **Error! No sequence specified.** HDFS

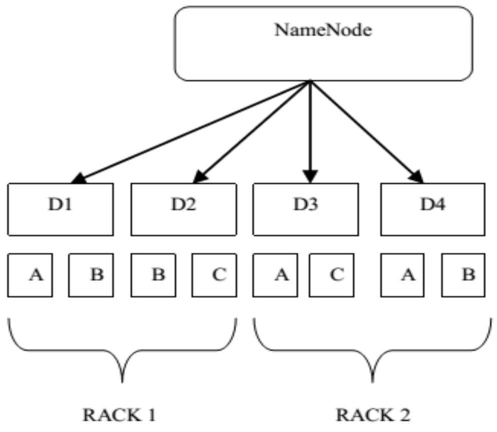

Figure 2: Hadoop MapReduce

The Hadoop MapReduce framework consists of one Master node termed as JobTracker and many Worker nodes called as TaskTrackers. The user submitted jobs are given as input to the JobTracker which transforms them into a numbers of Map and Reduce tasks. These tasks are assigned to the TaskTrackers. The TaskTrackers scrutinizes the execution of these tasks and at the end when all tasks are accomplished; the user is notified about job completion. HDFS provides for fault tolerance and reliability by storing and replicating the inputs and outputs of a Hadoop job.

Hadoop framework is popular for HDFS and MapReduce. The Hadoop Ecosystem also contains different projects which are discussed below [7] [8]:

- *Apache HBase:* A column-oriented, non-relational distributed key/value data store which is built to run on top of the HDFS platform. It is designed to scale out horizontally in distributed compute clusters.
- *Apache Hive:* A data warehouse infrastructure built on top of Hadoop for providing data summarization, querying, and analyzing large datasets stored on Hadoop files. It is not designed to offer real-time queries, but it can support text files, and sequence files.
- *Apache Pig:* It provides a high-level parallel mechanism for the programming of MapReduce jobs to be executed on Hadoop clusters. It uses a scripting language known as Pig Latin, which is a data-flow language geared towards processing of data in a parallel manner.
- *Apache Zookeeper:* It is an Application Program Interface (API) that allows distributed processing of data in large systems to synchronize with each other in order to provide consistent data to client requests.
- *Apache Sqoop:* A tool designed for efficiently transferring bulk data between Hadoop and structured data stores such as relational databases.
- *Apache Flume*: A distributed service for collecting, aggregating, and moving large amount of log data.

III. Applications Of Hadoop

Few of the applications of Hadoop are given below [9] [10] [11]:

- Log and/or clickstream analysis of various kinds
- Marketing analytics
- Online travel booking
- Energy discovery and energy savings
- Infrastructure management
- Fraud detection

- Health care
- Tracks various types of data such as Geo-location data, machine and sensor data, social media data.

IV. Job Scheduling

Scheduling in Hadoop uses different kinds of scheduler algorithms. The earliest versions of Hadoop used default FIFO scheduler. Then Facebook and Yahoo after considerable efforts in this area came up with Fair Scheduler and Capacity Scheduler respectively. These were then added to the later versions of Hadoop.

A. The Default FIFO Scheduler

The earlier versions of Hadoop used very straightforward approach for dealing with user's jobs. They were scheduled to run in the order they were submitted i.e., FIFO (First in First Out) principle [12],

After some time, assigmnent of priority to the jobs was provisioned as a new facility which is optional. Priority makes the job scheduler to choose the next job that is of highest priority. This option is not turned on by default but can be availed if required. But with FIFO scheduler preemption is not supported along with priority. So there is a chance that a long running low priority job may end with still blocking a later on scheduled high-priority job. Manually modifying job priorities in the FIFO queue certainly does work, but it requires active monitoring and management of the job queue. The problem with FIFO scheduler is that Hadoop dedicates the entire cluster to each job being executed. Hadoop offers two additional job schedulers that take a different approach and share the cluster among multiple concurrently executing jobs. The Capacity Scheduler and Fair Scheduler give a more sophisticated way of managing cluster resources across multiple concurrent job submissions. These schedulers are discussed in following subsections.

B. Fair Scheduler

Fair scheduling [13] is a method of allocating resources to jobs such that all jobs on an average get an equal share of resources over time. The Fair scheduler is intended to give a fair share of cluster capacity over time. If only one job is running, it has the privilege to access all the capacity of cluster. As users submit more jobs, free task slots are shared among each user in a manner it gives fair share of the cluster. The fair scheduler is fair enough with both shorter and longer jobs in the way that it lets shorter jobs to finish in a reasonable time while not starving longer jobs. This also allows for multiple users to share cluster in an easy manner. Fair sharing can also work with jobs assigned priorities.

The Fair Scheduler arranges jobs into pools and allocates resources fairly among these pools. Each user is assigned a pool by default which allows for equal sharing of the cluster. Inside each pool either FIFO or fair sharing scheduling would have been employed. Within each pool, the default model is to share the pool across all jobs submitted to that pool. Therefore, if the cluster is split into pools for two users say user A and user B, each of whom submit three jobs, the cluster will execute all six jobs in parallel. Suppose a pool has not received its fair share for some time period, then the scheduler optionally supports pre-emption of jobs in other pools. The Scheduler will be allowed to kill tasks in pools running over capacity so that slots can be given to the pools running under capacity. In order to guarantee that not to starve "production" jobs, the pre-emption can be used while still allowing the Hadoop cluster to be used by research and experimental jobs.

C. Capacity Scheduler

Capacity scheduler [14] [15] lias been designed specifically for those enviromnents where there is need for fair sharing of computation resources among large number of users. It takes a slightly different approach to multiuser scheduling. A cluster is made up of a number of queues, which may be hierarchical and each queue has a capacity allocated to it. Within each queue jobs are scheduled using prioritized FIFO scheduling. In effect, the capacity scheduler allows users or organization to simulate a separate MapReduce cluster with FIFO scheduling for each user or organization.

D. Improvements to Schedulers based on Speculative Execution of Tasks

Sometimes it may happen that few tasks in a set of tasks of a job continue to work slowly which will cost to the overall job execution time. Because of these straggling tasks in a job can make whole job to take much more time to finish then it would have finished in less amount of time only. There may be various reasons like high load on a CPU of a node, slow running of a background processes, software miss-configuration or hardware degradation. Hadoop attempts to detect and consequently launches another backup of a task that runs slower than expected time. This is called as speculative execution of tasks. If the original task finishes before the speculative task, then the speculative task is killed, if the speculative task finishes first, then the original is killed. Speculative execution [16] does not ensure reliability of jobs. If there are bugs in original task that sometimes hangs the task then the same bugs appears in the speculative task. So in this type of situation it's unwise to go for a speculative task. So there is a need to fix the bug so that the task does not slow down.

LATE Scheduler*:* Longest Approximate Time to End (LATE) is a considerable improvement over the default speculative execution. The LATE implementation of speculative scheduling relies implicitly on certain assumptions: a) uniform progress of tasks on node b) uniform computation at all nodes. But these assumptions break down easily in the heterogeneous clusters.

By considering a modified version of speculative execution instead of the progress made by a task so far, the computation of estimated remaining time is done which gives a more clear assessment of straggling tasks impact on the overall job response time.

Delay Scheduler*:* Fair Scheduler is designed that aims to assign fair share of capacity among all the users. It suffers from two locality problem. The first is head-of-line scheduling that occurs in small jobs. The second locality problem is sticky slot problem

[17], To solve head of line problem, scheduler launches a task from a job on a node without local data to maintain fairness. When it is not possible to run a task on a node that does not contain data, then it is better to run task on some other node on the same rack. In delay scheduling, when a node requests a task, if the head-of-line job cannot launch a local task, it is skipped and looked at subsequent jobs. However, if a job lias been skipped long enough, non-local tasks are allowed to launch to avoid starvation.

Other Scheduling Methods: Along with these schedulers, there are other different types of schedulers [1] like Dynamic Priority Schedulers. As the name itself indicates, this scheduler supports capacity distribution dynamically among concurrent users based on priorities of the users. Deadline Constraint Scheduler lias been designed to address the issue of deadline but it focuses more on enhancing system utilization. When a job is submitted, it undergoes schedulability test that determines whether the job can be finished within the specified deadline or not. Availability of free slots is computed at the given time or in the future irrespective of all the jobs running in the system. Once it is determined that a given job can be completed within the given deadline that job is enlisted for scheduling. But the Deadline Constraint Scheduler discussed works in a non-preemptive manner. But instead jobs can be run in preemptive scheduling manner. This preemptive scheduling approach under a deadline [18] has some advantages like it avoids delay in production job while still allowing the system to be shared by other non-production jobs. A different type of schedulers discussed so far does not deal with resources availability at fine-grained basis. The Resource Aware Scheduler as the name itself indicates attempts to use the resources efficiently. It is still a research challenge.

V. A Look In To Technical Aspects

Hadoop lias some important configuration files [19] [20] that need to be considered while configuring the Hadoop. Few handful significant popular configuration files are given below with small description for each:

- hadoop-env.sh: Enviromnent variables used in scripts to run Hadoop.
- core-site.xml: Contains Configuration settings for Hadoop Core, such as I/O settings that are common to MapReduce and HDFS.
- hdfs-site.xml: Configuration settings for HDFS daemons: the namenode, the secondary namenode, and the datanodes.
- mapred-site.xml: Configuration settings for MapReduce daemons: the jobtracker, and the tasktrackers.
- masters: A list of machines that each run a secondary namenode.
- slaves: A list of machines that each run a datanode and a tasktracker.

VI. Yet Another Resource Negotiator (YARN)

Hadoop is one of the widely-adopted cluster computing frameworks for processing of the Big Data. Although Hadoop arguably has become the standard solution for managing Big Data, it is not free from limitations. MapReduce lias reached scalability limit of 4000 nodes [21], Another limitation is Hadoop's inability to perform fine-grained resource sharing between multiple computation frameworks. To solve these limitations, the open source community proposed the next generation MapReduce called YARN (Yet Another Resource Negotiator) [21] [22],Computer scientists and engineers are trying hard to eliminate these limitations and improve Hadoop. YARN eliminates scalability limitation of the first generation MapReduce paradigm.

The earlier version of Hadoop did not have YARN but it was added in the Hadoop 2.0 version to increase the capabilities [23], A more general processing platform is provided by YARN-based architecture of Hadoop 2.0 that is not limited to MapReduce. YARN'S basic idea is to split up the two major functionalities of the JobTracker, resource management and job scheduling into separate daemons. The idea is to have a global ResourceManager and per- application ApplicationMaster. The ResourceManager arbitrates resources among all the applications in the system and it lias two components: Scheduler and Applications Manager.

The Scheduler is responsible for allocating the resources among running applications. The ApplicationsManager accepts job-submissions, negotiating the first container for executing the application and provides the service for restarting the ApplicationMaster container on failure. The resource management abilities that were present in MapReduce are also acquired by YARN which tones up MapReduce in processing the data more efficiently. With YARN multiple applications can run in Hadoop all sharing a common resource management.

A. Comparison of YARN and MapReduce

By separating resource management functions from the programming model, YARN delegates many scheduling- related functions to per-job components. In this new context, MapReduce is just one of the applications running on top of YARN. This separation provides a great deal of flexibility in the choice of programming framework. Programming frameworks running on YARN coordinates intra-application communication, execution flow, and dynamic optimizations as they see fit, unlocking dramatic performance improvements. The Classic Hadoop and YARN architectures use a different scheduler. Classic Hadoop uses a JobQueueTaskScheduler, while YARN uses Capacity Scheduler by default.

It has some significant differences from old MapReduce.

- Introduction of Resource Manager responsible for managing and assigning global cluster resources.
- Introduction of Application Master in each of the application. The application master interacts with resource manager to request compute resources.
- Introduction of Node Manager responsible for managing user processes per node.

In earlier version of Hadoop the JobTracker was closely tied with MapReduce framework. It was responsible for both resource management and application management. JobTracker would allow running Hadoop MapReduce jobs only. The new resource manager allows running other services like MPI within the same cluster via Application Master. In old Hadoop map and reduce slots could not be interchangeably used. This would mean that the cluster would be largely underutilized during a pure map or reduce phase. In the newer avatar of Hadoop slots can be reused as there is much better resource utilization.

VII. CONCLUSION

The paper begins with a brief introduction about Big Data. Big data can bring valuable benefits to the business. Then the paper discusses about one of the technologies that handle the Big Data, the Hadoop. Then paper talks about HDFS and MapReduce programming model. Then we talk about some applications of Hadoop. Then the various schedulers used in Hadoop are discussed in brief. We look into technical aspects of Hadoop where some important configuration files of Hadoop are discussed. Since MapReduce undergoes some significant limitations if the numbers of nodes are increased, the technology that overcomes this limitation is YARN which is briefly discussed.

REFERENCES

[1] B.Thirumala Rao, Dr.L.S.S.Reddy, "Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments", International Journal of Computer Applications (0975 - 8887) Volume 34- No.9, November 2011.

[2] Kala Karun. A Chitharanjan.K, "A Review on Hadoop - HDFS Infrastructure Extensions", Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013).

[3] Lizlie Wang, Jie Tao, Rajiv Ranjan , Holger Marten, AchimStreit, Jingying Chen, DanChen, "G-Hadoop: MapReduce across distributed data centers for data-intensive computing", 2013.

[4] WeijiaXu, Wei Luo, Nicholas Woodward, "Analysis and Optimization of Data Import with Hadoop", 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum

[5] HUANG Lan. WANG Xiao-wei. ZHAI Yan-dong. YANG Bin. "Extraction of User Profile Based on the Hadoop Framework".

[6] Jeffy Dean.Sanjay Ghemawat. MapReduce, "Simplified Data Processing on Large Clusters", OSDI04: Sixth Symposium On Operating System Design and Implemention, Ssn Francisco, CADecember, 2004.

[7] Introduction to the Hadoop Software Ecosystem. Other applications. 10 ways companies are using Hadoop (for more than ads). Business Applications of Hadoop.

[8] Hadoop: The Definitive Guide, Third Edition, Textbook. Hadoop's Fair Scheduler.

[9] Jagmohan Chauhan, Dwiglit Makaroff and Winfried Grassmann, "The Impact of Capacity Scheduler Configuration Settings on MapReduce Jobs".

[10] M. Zaharia, A. Kowinski, A. Joseph, R. Katz and I. Stoica, Improving mapreduce performance in heterogeneous environments. USENK OSDI. 2008

[11] DongjinYoo, Kwang Mong Sim, "A comparative review of job scheduling for mapreduce," Multi-Agent and Cloud Computing Systems Laboratory, Proceedings of IEEE CCIS2011.

[12] Premptive Hadoop Jobs Scheduling under a Deadline, "Li Liu, Yuan Zhou, Ming Liu, Guandong Xu, Xiwei Chen, Dangping Fan, Qianra Wang", IEEE 2012 Eighth International Conference on Semantics, Knowledge and Grids.

[13] Hadoop Administrative Guide.

[14] Hadoop Cluster Configuration Files.

[15] Arinto Murdopo, Jim Dowling, "Next Generation Hadoop: High Availability for YARN".

[16] Vinod Kumar Vavilapalli, Aran C Murthy, Chris Douglas, SharadAgarwal, MahadevKonar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, BikasSaha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin, Reed, Eric Baldeschwieler, "Apache Hadoop YARN: Yet Another Resource Negotiator".