Scheduling Jobs in Hadoop MapReduce framework for Heterogeneous Environments

Ankita Nitin Saxena*
Department of Computer Science
SRMS College of Engineering Technology & Research
Bareilly 243202, U.P., India.
*shaily.ankita@gmail.com

Abstract

Job scheduling algorithm the foremost technology in the Hadoop platform, is used to control the sequence of task execution and designate user's job for efficient usage of resources. The concept which was fabricated for homogeneous cluster environments, is now more often used for multiple heterogeneous environment. This Variation puts a toll on the performance of Hadoop Scheduler as a fixed threshold is used for job selection. The aim of this paper is to provide the comparative study of various default and improved job scheduling algorithms in Hadoop and this will help the researcher to know the recent development in Job scheduling algorithm and also some new research directions in heterogeneous environment. It will help them to choose best algorithm for their particular application.

Keywords- Hadoop, MapReduce, Scheduling Algorithm, Heterogeneous Environment.

Introduction

Hadoop, a popular open-source implementation of the Google's MapReduce model is primarily developed by Yahoo. Yahoo servers generate hundreds of terabytes data for at least 10,000 cores of processors and it uses Hadoop for scheduling the tasks that are already too crowded. Hadoop is also used by Facebook which needs to process more than 20 terabytes of new entries per day. Besides various other websites are using Hadoop to process large data and requests on per day basis. Not only the web based applications but the scientific based applications widely use Hadoop for various tasks such as seismic and natural language simulations etc.

Its MapReduce exhibits several advantages that differ from those of traditional parallel computing systems. First, regarding scalability, even when new machines are added to a cluster, the system still works well without reconstruction or much modification. Second, regarding fault tolerance, the MapReduce model can automatically manage failures and mitigate complexity of fault tolerance mechanisms. As there is more than one machine so on failure of one node some other nodes can be utilized for the aborted task due to machine failure Thirdly This model does not have the need to learn programming for parallel and distributed environments. A program executed using the MapReduce model partitions jobs into a numerous task to be assigned and run on multiple nodes in the cluster, and the program collects the processing results of each node to be return. MapReduce scheduler just considers scheduling in homogeneous environment, and fails to find tasks which prolonging execution time.

The goal of this paper is to perform the complete study of basic and improved algorithms of Hadoop MapReduce in heterogeneous environment. There are 4 sections in the paper.2nd section deals with the key concept of MapReduce and its architecture and Section 3 explain default scheduling algorithm in Hadoop MapReduce and also described some of improved scheduling algorithms for heterogeneous environment and in section 4 concludes the paper with some future research direction.

Hadoop

Apache Hadoop is used for storing and processing large data set clusters which is an open source software framework, which is based on java programming framework. As Hadoop provides Cheap and reliable storage, it can be used more efficiently. Its library can easily detect and handle failures at the top most application layer and so can provide reliable service to cluster of computers. So this platform of Hadoop has enabled large applications to be run on thousands of computers and massive data without affecting too much on allocation and distribution of data and calculation. Hadoop is much more than a highly available, massive data storage engine. The leading benefit of using Hadoop is that one can easily utilize data processing and storing feature. Hadoop using HDFS for data storing and using MapReduce to processing that data.

Mapreduce

It's the model for programming that can be used as the combination of large nodes and computing resources as a cluster to parallel utilize the resources and large data sets. It was the proposal of Google in 2004.Here single application to be executed is termed as "job". A job can be divided into two parts: first is "map" and second is "reduce", in which the map-tasks run the map function and the reduce-tasks run the reduce function. Map function processes input data assigned by the master and produce many intermediate key, value. The value of pair (key, value) generated during mapping function used by the reducer for merging, sorting and returning result. The model being discussed works on the policy of divide and conquer. Large data sets are first distributed to nodes for parallel processing, so that both its execution time and its performance are optimized.

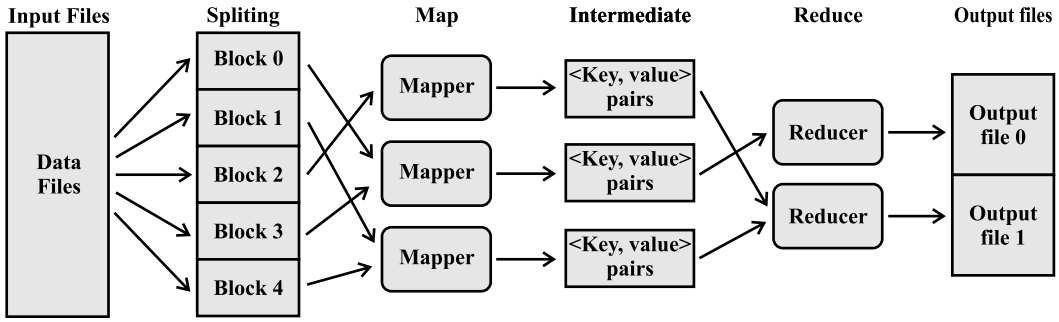

Figure 1: The overview of the MapReduce data flow

During function run, input data is fragmented to data blocks of same size, and then for parallel processing they are assigned to nodes in the cluster. After the map function is performed, the generated output is an

intermediate datum composed of several key, value. The nodes that perform the reduce function obtain these intermediate data, and finally generate the output data. Figure 1 shows the MapReduce Data flow. When executing jobs, the Job Tracker manages all jobs scheduling and task distributions, and each Task Tracker is responsible for performing tasks and designate the output to the Job Tracker. Job-Tracker and Task Trackers employ the sending of heartbeat message for communication and data transferring. Each Task Tracker has a different number of task slots, which are usually based on the performance of nodes, to set the number of task slots which can one by one process tasks. As some slots becomes empty and there are more jobs to be done heartbeat messages are used for sending and for responses to and from Job Trackers to Task Tracker to perform the tasks. The same heartbeat, as soon as the tasks get over the same heartbeat message is sent by Task Tracker to inform the Job Tracker about the task completion.

Its benefit lies in its ease to use. This model also provides computing transparency. In this system mapper and reducer nodes are easily segregated for respective computing. When programming, a programmer does not need to spend extra time on data and program division. Therefore, a programmer does not need to have a deep understanding of parallel computing. A programmer must simply focus on the normal function processing rather than the parallel processing. This can simplify the application development process substantially and shorten the development time. In recent years, MapReduce has been used increasingly more in large-scale distributed systems. For example, Google uses MapReduce to manage large amounts of data daily.

HDFS- Distributed File System

This file system is given by Google and helps to give an efficient and reliable data access for large servers. File size is 64 MB, used for read and rarely used for overwriting, Google File System (GFS) is made such that data is stored and arranged as optimized data centers and they helps to give high throughput, low latency and they can also sustain nodes crash.

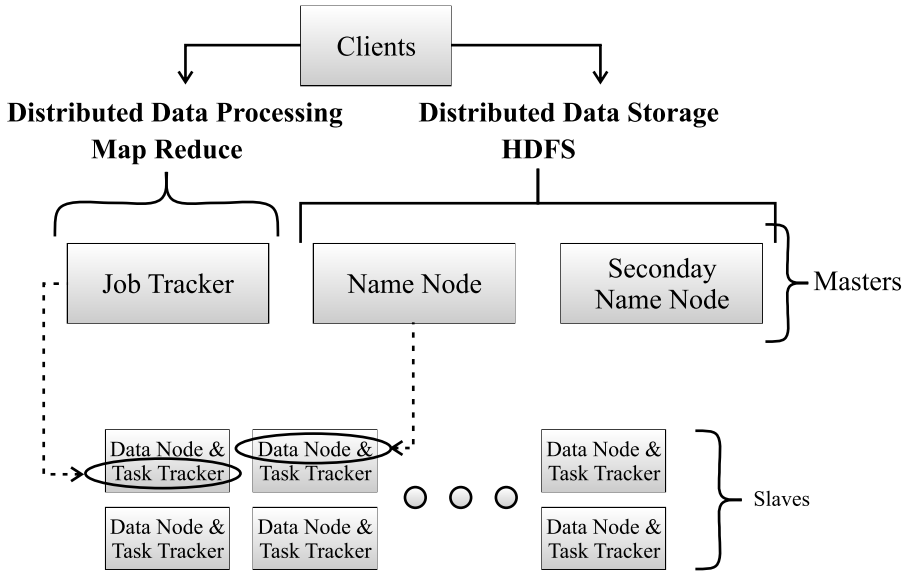

Figure 2: Hadoop Distributed File System

HDFS replicates data amongst various centers for reliability .and hence computation also becomes fast. To enable Data access worldwide over web, from different clients it is also provided over HTTP. HDFS uses the master/slave configuration. According to shown in Figure 2 HDFS has one Name-Node and n Data-Nodes per cluster. The role of Name-Node is mapping of data blocks to Data-Nodes and managing file system operations like file opening, file closing and file renaming and also of directories data. As per the instructions of Name Node, Data Nodes Creates blocks, delete and replicates block of data. There is an additional namespace in the Name Node which is used for recording user actions on files of creation, deletion and modification. Name Node takes the decision of data blocks replication.

Scheduling Algorithm used in Hadoop

The default Scheduling algorithm is based on FIFO where jobs were executed in the order of their submission. Afterwards priority setting were also included. Lot of contribution came from Facebook and Yahoo for schedulers development i.e. Fair Scheduler and Capacity Scheduler respectively which subsequently released to Hadoop Community.

Default FIFO Scheduler

This scheme utilizes the concept of a queue which executes as a FIFO queue. Once the job is fragmented in to individual tasks they are assigned to free slots in the queue from where they are available on task tracker nodes. Though priorities can be assigned, but they are not used in default scheduling. Here a job utilizes the whole cluster so new jobs had to wait for an empty cluster in FIFO order of job lining up. Though clusters can be shared but they are not as sharing might pose more vulnerable problems. Such techniques are usually done in case of productions where timely job submission is very important as the users here make adhoc queries.

Fair Scheduler

The Fair Scheduler product made at Facebook is one which can be used for managing tasks in the clusters of Hadoop. The Fair Scheduler aims to give every user a fair share of the cluster capacity over time. Pools are used by Users for job assignment, where each pool is given some limited map and reduce slots as the minimum limit. Free slots in idle pools may be allocated to other pools, while excess capacity within a pool is shared among jobs. This technique uses preemptive scheduling so that the starving pools can be given their shares under capacity Here priorities are also being used and pools are given fair chance according to highest priorities. Task Tracker slots used for processing and task allocation, acts as the scheduler who tracks the lack amongst actual utilized time and ideally fair allocated share of the job. The tasks are assigned or allocated in the freely available slots as per the highest time deficit. So, on an average all jobs get almost equal share of the resources. Shorter jobs finish soon so they are given less resources as compared to the longer jobs which are assigned resources so that they are not starved of resources.

Capacity Scheduler

Capacity Scheduler (Thirumala *et al*., 2011) the idea coined by Yahoo focuses on applications where users are large, and the computing resources have to be allocated amongst multiple users. This technique of scheduling jobs is done by allocating resources to users which submit configurable slots of mapping and reducer data. Scheduling queues with already allocated jobs are configured according to their capacity they hold and free slots are shared by other queues. This queue operates as per the modified scheduling priority based technique where highest is given to oldest in time job and least to the new task submitted, As and when the Task Tracker has a free slot lowest load holding queue is selected from which the task which is the oldest is chosen. A task is then scheduled from that job.

Scheduling improvements for heterogeneous environment

Lot of improvements are being done on scheduling policies in Hadoop. The latest work being done is on the Delay Scheduler, Dynamic Proportional Scheduler (MateiZaharia *et al*., 2010] which has given differentiated service in applications using Hadoop jobs which helps to optimize the priority levels assigned to their jobs. However, this does not guarantee that the job will be completed by a specific deadline. Deadline Constraint Scheduler (Sandholm *et al.,* 2010) scheme helps to increase system utilization as per the deadline. The Schedulers described above attempt to allocate capacity fairly among users and jobs, they make no attempt to consider resource availability on a more fine-grained basis. Resource Aware Scheduler (Anyanwu *et al.,* 2010) considers the resource availability to schedule jobs. In the following sections, we compare and contrast the work done by the researchers on various Schedulers.

Longest Approximate Time to End (LATE) - Speculative Execution

Systems progress slowly due to several reasons like–high CPU load on the node, slow background processes etc. All tasks should be finished for completion of the entire job. So in this scheme the scheduler finds the slowest running task, which replaces it with another equivalent task that can fulfill its job and run fast. This is speculative execution of tasks. If the backup copy completes faster, the overall job performance is improved. This approach might be an optimization still it does not ensure job reliability. If there are errors or faults, then this approach might replicate the problem. Such errors should be removed for efficient system run. Certain assumptions are made for the scheme: a) Uniform Task progress on nodes b) Uniform computation at all nodes. So, it can be obtained that this method is well suited for homogeneous clusters in real life applications. (Zaharia *et al*., 2008) gave a modified proposal for speculative execution termed as Longest Approximate Time to End (LATE) algorithm which uses another parameter for scheduling tasks. So, this approach utilizes the concept of remaining time which helps to identify struggling tasks and hence help to improve overall response time.

Self-Adaptive MapReduce (SAMR)

LATE scheduling algorithm does not compute the remaining time for tasks correctly, and cannot find real slow tasks in the end. To address this limitation, (Chen *et al.*, 2010) propose a Self-Adaptive MapReduce scheduling algorithm which is another approach for heterogeneous environments. The SAMR

scheduling algorithm improves MapReduce by saving execution time and system resources. It defined fast nodes and slow nodes to be nodes which can finish a task in a shorter time and longer time than most other nodes. In heterogeneous clusters, nodes require different times to accomplish the same tasks due to their differences, such as computation capacities, communication, architectures, memory, and power. SAMR scheduling algorithm could be further improved in term of in three aspects. First, it will focus on how to account for data locality when launching backup tasks, because data locality may remarkably accelerate the data load and store. Second, it considering a mechanism to incorporate that tune the parameters should be added. At last, it will be evaluated on various platforms by first evaluated on rented Cloud Computing platform (Chen *et al.,* 2010).

Delay Scheduling

This scheme identified two locality problems – head-of-line scheduling and sticky slots. First locality problem happens in small jobs (less data to read). The issue is that always the task at the head of the line is by mistake allotted to next slot, no matter on which this slot lies. Head-of-line scheduling problem was for mostly happened at Facebook in the file system of Hadoop which didn't had delay scheduling. Another locality problem, sticky slots, is tendency of a job to be assigned to same slot repeatedly.

To rectify the Head of line problem, scheduler started a task from a job on the local node for efficient fairness. Running on a node that contains the data (node locality) is most efficient, but when this is not possible, running on a node on the same rack (rack locality) is faster than running off-rack. 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 such as non-local tasks used to avoid starvation.

Deadline Constraint Scheduler

Deadline Constraint Scheduler (Mark *et al*., 2009) focused on the deadline issue but tries to increase more of the system utilization. Estimation model determines the available slot based a set of assumptions:

- All nodes are homogeneous nodes and unit cost of processing for each map or reduce node is equal
- Input data is distributed uniform manner such that each reduce node gets equal amount of reduce data to process
- Reduce tasks starts after all map tasks have completed;
- HDFS.has the required input to be used.

Arranging of tasks is done by cost estimation and priority based approaches. If the available tasks have the desired deadline, then only they are scheduled. After which the queues are tested whether they complete in the specified time or not. Free slots availability is computed at the given time or in the future irrespective of all the jobs running in the system. The job is enlisted for scheduling after it is determined that the job can be completed within the given deadline. Tasks can be scheduled only if they are less than both map and reduce. This Scheduler also emphasis that when a deadline for job is different, they are given to Task Tracker which ensures that the specified deadline is met.

Resource Aware Scheduling

Two possible resource-aware Job Tracker scheduling mechanisms are: 1) Dynamic Free Slot Advertisement-Instead of having a fixed number of available computation slots configured on each Task Tracker node, this number is computed dynamically using the resource metrics obtained from each node. In one possible heuristic, overall resource availability is set on a machine to be the minimum availability across all resource metrics. In a cluster that is not running at maximum utilization at all times, this is expected to improve job response times significantly as no machine is running tasks in a manner that runs into a resource bottleneck. 2) Free Slot Priorities/Filtering- In this mechanism, cluster administrators will configure maximum number of compute slots per node at configuration time. Resource availability marks the order in which free Task Tracker slots are advertised. As Task Tracker slots become free, they are buffered for some small-time period (say, 2s) and advertised in a block. Task Tracker slots with higher resource availability are presented first for scheduling tasks on. In an environment where even short jobs take a relatively long time to complete, this will present significant performance gains. Here Job response time needs to be improvised rather than scheduling tasks onto the next available free slot onto a resource-rich machine, even if such a node takes a longer time to become available. Buffering the advertisement of free slots allowed for this scheduling allocation.

Conclusion and Future Work

In this paper, I have described the overview of Hadoop MapReduce and their scheduling issues. Then, some comparative study of default scheduling algorithms as well as improved scheduling algorithm for heterogeneous environment in Hadoop is done. After this analysis of these (SAMR, late etc.) algorithm is performed on basis of parameters like Idea to implementation, advantages and disadvantages. It is observed that we need universal scheduling algorithm to get best result from MapReduce model in heterogeneous environment.

References

Ghemawat, S., Gobioff, H., Leung, S.T. 2003. The Google file system. In *ACM SIGOPS operating systems Review*. vol. 37(5), 29-43.

Jeffrey, D., Ghemawat, S. 2008. MapReduce: simplified data processing on large clusters. *Communications of the ACM* . 51(1),107-113.

Kamal, K.., Anyanwu, K.. 2010. Scheduling hadoop jobs to meet deadlines. In *Cloud Computing Technology and Science (CloudCom), IEEE Second International Conference.* 388-392.

Mark, Y., Garegrat, N., Mohan, S. 2009.Towards a resource aware scheduler in hadoop. In *Proc. ICWS*.102-109.

Matei, Z., Konwinski, A., Joseph, A.D., Randy, Katz, H., Stoica I. 2008. Improving MapReduce Performance in Heterogeneous Environments. .In *OSDI* .8(4),4-7.

Quan, C., Zhang, D., Guo, M., Deng, Q., Guo, S. 2010. Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*. 2736-2743.

Rao K.T., Reddy S. 2012.Survey on improved scheduling in Hadoop MapReduce in cloud environments. *arXiv preprint arXiv:1207.0780*.

Rao, K. T., P. Sai Kiran., Reddy S., Reddy K. 2010 .Genetic Algorithm For Energy Efficient Placement Of Virtual Machines In Cloud Environment. In proc IEEE International Conference on Future Information Technology (IEEE ICFIT), China.

Rao, K.T., Sridevi, N.V, Krishna Reddy, V., Reddy, L.S. 2010.Performance issues of heterogeneous hadoop clusters in cloud computing." *arXiv preprint arXiv:1207.0894* .

Rob, P., Dorward, S., Griesemer, R., Quinlan, S. 2005. Interpreting the data: Parallel analysis with Sawzall. *Scientific Programming* 13(4). 277-298.

Reddy, V., Krishna, B., Thirumala, R., Reddy, S. 2010.Research issues in cloud computing. *Global Journal of Computer Science and Technology.* 11(11). 32-40.

Salma, K.., Sameh, A., Nassar, S., Elsayed, M. 2013. Mapreduce performance in heterogeneous environments: A review. *International Journal of Scientific & Engineering Research* . 4(4) . 410-416.

Thomas, S., Lai K. 2010. Dynamic proportional share scheduling in hadoop. In *Workshop on Job Scheduling Strategies for Parallel Processing*. 110-131.