Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 12, December 2015, pg.171 – 176

Hadoop MapReduce Scheduling Algorithms – A Survey

Ms. Anjana Sharma

Senior Assistant Professor, Computer Science and Engineering Department, New Horizon College of Engineering, Bangalore, India

anjana.nhc@gmail.com

Abstract - Hadoop framework has been widely used to process large-scale datasets on computing clusters. In Hadoop, MapReduce framework is a programming model which processes terrabytes of data in very less time. This framework uses a task scheduling method to schedule task. There are various methods available for scheduling task in MapReduce framework. Scheduling of jobs in parallel across nodes is a major concern in distributed file system. The objective of the research is to study MapReduce and analyze different scheduling algorithms that can be used to achieve better performance in scheduling.

Keywords- Big Data, Hadoop, MapReduce, Scheduling, Distributed data processing

I. INTRODUCTION

1.1 *Overview*

Due to the development of new technologies, devices, and communication means like social networking sites, the amount of data produced by mankind is growing rapidly every year. Big data is a collection of large datasets that cannot be processed using traditional computing techniques. Processing large-scale datasets has become an increasingly important and challenging problem as the amount of data created by online social networks, healthcare industry, scientific research, etc., explodes. This has certain challenges, it is very difficult to transfer such a huge data to the

computing node and very high speed networks will be needed. Also the cost required to transmission of data is also high. The era of big data requires new ways to store, manage, access and process the colossal amount of available data. Since parallel processing is scalable and can gain performance improvement by several orders of magnitude, it is generally accepted as a must for data-intensive applications in the big data era.

Hadoop

Hadoop is an Apache open source framework that allows distributed processing of large datasets across clusters of computers using simple programming models. A Hadoop frame-worked application works in an environment that provides distributed storage and computation across clusters of computers. The core of Hadoop consists of a storage part, Hadoop Distributed File System (HDFS) and a processing part called MapReduce. The Hadoop File System (HDFS) has a master/slave architecture. The master process (called as NameNode) manages the global name space and controls the operations on files. Each slave process (called as DataNode) stores the files in form of data blocks and performs operations as instructed by the NameNode. The NameNode manages the data replication and placement for fault-tolerance, performance and reliability. The NameNode splits and stores files in 64MB data blocks across the DataNodes. Usually 3 replicas of each data block are stored in the HDFS. Failure detection mechanism is implemented in form of regular heartbeats from DataNodes to NameNode. If there is no heartbeat from a DataNode for long time it is marked as failed not used for further operations and if required additional duplicates of its data are created.

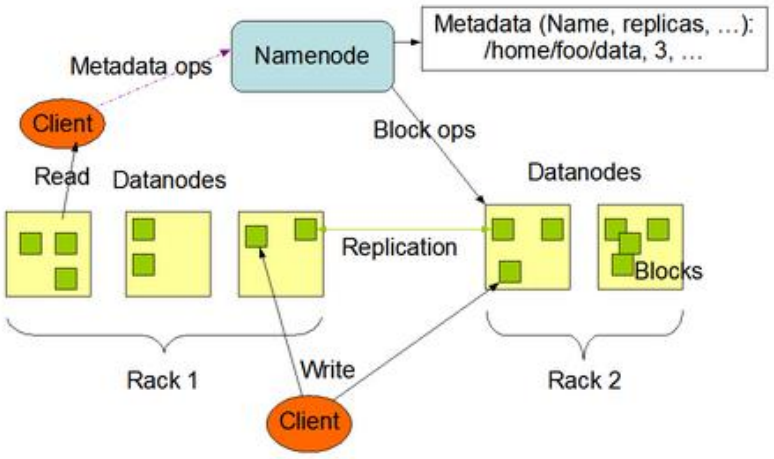

Fig. 1 HDFS Architecture

MapReduce

MapReduce is a simple yet powerful programming model designed for processing large scale datasets in a distributed and parallel fashion. Originally developed by Google, and later on popularized by its open-source implementation

Hadoop , MapReduce has been widely used in practice by companies including Google, Yahoo!, Facebook, Amazon, and IBM. A data processing request under the MapReduce framework, called a job, consists of two types of tasks: map and reduce. A map takes a set of data and processes it to produce intermediate results (key-value pairs). Then, reduce tasks fetch the intermediate results and carry out further computations to produce the final result. Map and reduce tasks are assigned to the machines in the computing cluster by a master node that keeps track of the status of these tasks to manage the computation process. The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes.

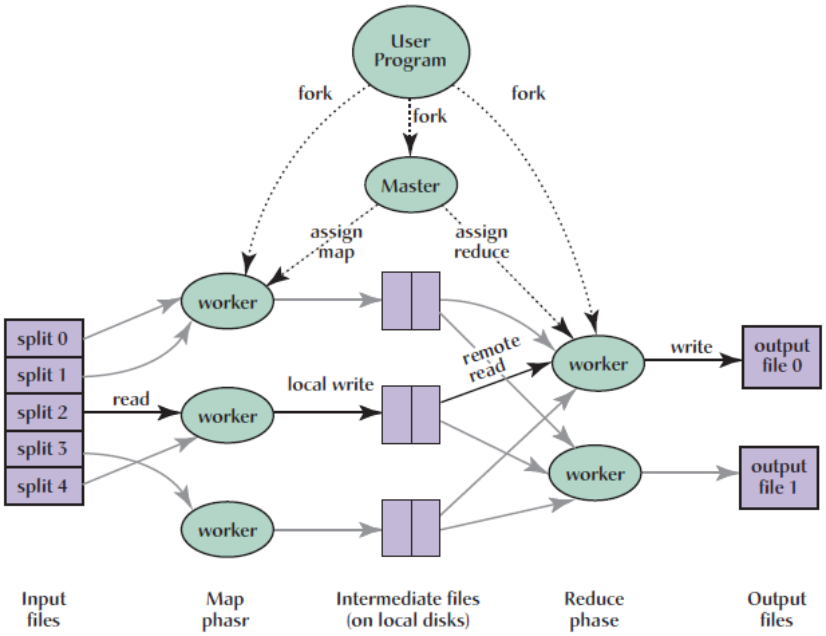

Fig. 2 Hadoop/MapReduce Overview

II. MAPREDUCE SCHEDULING

There are three important scheduling issues in MapReduce such as locality, synchronization and fairness. Locality is a very crucial issue affecting performance in a shared cluster environment, due to limited network bisection bandwidth. Synchronization of these two phases (Map and Reduce phase), made up essential help to get overall performance of MapReduce Model. A map-reduce job with a heavy workload may dominate utilization of the shared clusters, so some short computation jobs may not have the desired response time. . The demands of the workload can be elastic, so fair workload to each job sharing cluster should be considered. To solve these important issues many scheduling algorithms have been proposed in the past decades. in the next section we describe these algorithms.

2.1 ***FIFO Scheduler***

FIFO is the default Hadoop scheduler. The main objective of FIFO scheduler to schedule jobs based on their priorities in first-come first-out of first serve order. The FIFO scheduler operates using a FIFO queue. Job is divided into independent tasks and then they are put into the queue and allotted to free slots as they get acquirable on TaskTracker nodes. Job Tracker pulls oldest job first from job queue and it doesn't consider about priority or size of the job. FIFO scheduler have limitations like poor response times for short jobs compared to large jobs, Low performance when run multiple types of jobs and it give good result only for single type of job. to address these problems scheduling algorithms such as Fair and Capacity was introduced.

2.2 ***Fair Scheduler***

This mechanism was developed at Facebook to manage access the Hadoop cluster. The objective of this scheduler is to assign each user a fair share of the cluster capacity over a time. Users may assign jobs to pools, with each pool allocated a guaranteed minimum number of Map and Reduce slots. Free slots in ineffective pools may be allocated to new pools . Its preemptive technique that means the scheduler will kill tasks in pools running over capacity in order to give the slots to the pool running under capacity. Priority criteria are also assigned to various pools. Here tasks are scheduled in interleaved manner based on priority.

2.3 ***Capacity Scheduler***

Capacity Scheduler was developed at Yahoo. It 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. It has a design similar to Fair scheduler. But this uses queues instead of pool. Each queue is assigned to an organization and resources are divided among these queues. The Queues that hold jobs are bestowed their organized capacity. If a queue has heavy load, it seeks unallocated resources, then makes redundant resources allocated evenly to each job. For maximizing resource utilization, it allows re-allocation of resources of free queue to queues using their full capacity. When jobs arrive in that queue, running tasks are completed and resources are given back to original queue. It also allows priority based scheduling of jobs in an organization queue. This scheduling algorithm takes care of FIFO's problem of low utilization of resources.

2.4 ***Longest Approximate Time to End (LATE)***

It is not uncommon for a particular task to continue to progress slowly. This may be due to several reasons like high CPU load on the node, slow background processes etc. All tasks should be finished for completion of the intact job.

Zaharia proposed LATE (Longest Approximate Time to End) scheduler to robustly improve performance by reducing overhead of speculation execution tasks. LATE scheduling algorithm tries to improve Hadoop by attempting to find real slow tasks by computing remaining time of all the tasks. LATE is based on three principles: prioritizing tasks to speculate, selecting fast nodes to run on, and capping speculative tasks to prevent thrashing. The advantage of the LATE algorithm is robustness to node heterogeneity, since only some of the slowest speculative tasks are restarted. This method does not break the synchronization phase between the map and reduce phases, but only takes action on appropriate slow tasks.

2.5 *Delay Scheduling*

This method performs well in Hadoop workloads because Hadoop tasks are short relative to jobs, and because there are multiple locations where a task can run to access each data block . When a node requests a particular task, if the head of line job can't assign local task, scheduler skips that task and starts looking for next jobs. However, if a job has been skipped long enough, we start allowing it to launch non- local tasks, to avoid starvation. This method improves problem of locality by asking jobs to wait for scheduling opportunity on a node with local data.

III. CONCLUSION

There is a huge amount of data lying in the industry so Hadoop has gained lot of importance. Hadoop can implemented on low cost hardware and can be used by many people on large number of dataset. MapReduce is the most important component in Hadoop. In this paper we have studied default hadoop schedulers for processing the tasks and taken into consideration their drawbacks and studied available schedulers. We can see that most of these schedulers discussed in this paper, address one or more problem. And choice of this scheduler for a particular job is up to the user.

REFERENCES

1. F Ahmad, S. Y. Lee, M. Thottethodi, T. N. Vijaykumar. *PUMA: Purdue MapReduceBenchmarks Suite*. ECE Technical Reports, 2012
2. G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica,Y. Lu, B. Saha, and E. Harris, Reining in the outliers in map-reduce clusters using mantri, in OSDI'10, pp. 1-16, 2010.
3. Apache Hadoop NextGen MapReduce (YARN). http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

4. C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in Proc. IEEE Int. Conf. Compute., Syst. Signal, Bangalore, India, Dec. 1984, pp. 175–179.
5. J. Chao, R. Buyya. MapReduce Programming Model for .NET-Based Cloud Computing.In Euro-Par'09, pp. 417-428, 2009.
6. Q. Chen, C. Liu, Z. Xiao, *Improving MapReduce Performance Using Smart Speculative Execution Strategy*. IEEE Transactions on Computer, 2013.
7. J. Dean and S. Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*, In OSDI'04, pp. 107-113, 2004
8. Radheshyam Nanduri, Niteshaheshwari, Reddy Raja,Vasudeva Varma, "Job Aware Scheduling Algorithm for MapReduce Framework", 3rd IEEE International Conference on Cloud Computing Technology and Science Athens, Greece.
9. Mark Yong, Nitin Garegrat, Shiwali Mohan: "Towards a Resource Aware Scheduler in Hadoop" in Proc. ICWS, 2009, pp:102-109
10. Jagmohan Chauhan, Dwight Makaroff and Winfried Grassmann, "The Impact of Capacity Scheduler Configuration Settings on MapReduce Jobs"
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. Hadoop's capacity scheduler: http://hadoop.apache.org/core/docs/current/capacity/sch eduler

