ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 3, March 2016

A Comparative Review on Different Scheduling Algorithms in Hadoop

Anuradha Sharma, Er. Sheenam Malhotra

M.Tech Student, Dept. of CSE, Sri Guru Granth Sahib World University, Punjab, India

Assistant Professor, Dept. of CSE, Sri Guru Granth Sahib World University, Punjab, India

ABSTRACT: In today's world large quantity of data is produced everyday by different business and scientific applications. Hadoop has become a dominant computation model for processing huge amount of information. Hadoop works on MapReduce framework which offers parallel processing to execute various data exhaustive jobs present on a cluster. With increase in use of hadoop, scheduling algorithms are required for efficient utilization of cluster resources and executing them in less time. This paper explains core components of hadoop to understand its working, some existing schedulers are compared to evaluate their performance and few parameters are proposed to improve the performance of hadoop.

KEYWORDS- Hadoop; MapReduce; HDFS; Scheduling

I. INTRODUCTION

In recent years, the data stored on web is growing considerably. The data is being generated and processed by various business and scientific applications such as web mail, search engines, machine learning, data mining, etc. This ever expanding data ranges from few gigabytes to several terabytes or even petabytes, referred to as big data. Big data is illustrated by 4V's: Variety, Volume, Velocity and Veracity. Variety means the kind of data, which can be structured, unstructured and semi-structured data including text, audio, video, sensor data, etc. Volume represents how large the data is, described in terabytes and petabytes. Velocity defines the movement and streaming of data. While,Veracity refers to reliability of the data.

Hadoop is the most popular tool offered today for storing and processing the huge volume of data in minimum time by many leading companies including Amazon, Google, Yahoo, Facebook, Microsoft, etc. It is an open source implementation of MapReduce framework, written in java mainly developed by Yahoo and currently maintained by Apache Foundation. It has been emerged as the favored solution to solve the issues disrupting conventional management and processing systems. The components of current Apache hadoop are Hadoop kernel, Map Reduce, HDFS, Hive, HBase, Zookeeper. The vital components are MapReduce and HDFS.

As the quantity of information increases, it creates problems for existing scheduling algorithms for processing. The main objective of scheduling algorithm is to manage load on cluster and to get minimum execution time with maximum utilization of system resources. In the remaining part of paper, Section II describes related work done in the field of scheduling in hadoop. Section III presents the technical background essential for understanding the execution of Hadoop. Section IV summarizes some of exiting scheduling algorithms and their comparison. Section V concludes the paper.

II. RELATED WORK

In [**1**] authors reviews different scheduling methods for handling issues of data locality and synchronization overhead. To dodge the synchronization overhead two approaches were discussed- asynchronous processing & speculative execution. For fairness constraint with data locality, delay scheduling with fair scheduler and quincy schedulers are described. The features, strengths and weaknesses were discussed among different approaches. The performance of MapReduce can be improved by eliminating these issues. In [**2**] a task scheduler is introduced for MapReduce framework that manages the performance of MapReduce tasks. The proposed scheduler predicts the performance of jobs dynamically and adjusts the allocation of resources for the jobs. The completion time of job is estimated

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 3, March 2016

dynamically during its execution by the task scheduler. Accurate predictions are not provided by this technique all the time but can be combined with dynamic scheduling policy for better results, that will allow fair management of completion times of multiple jobs.

To estimate the progress of tasks dynamically as well as adapts the varying environment as task overload occurs, Self-adaptive MapReduce scheduling algorithm (SAMR) [**3**] is proposed. SAMR improves MapReduce in terms of saving execution time and utilizing system resources. It incorporates historical information recorded on each node discovering slow tasks dynamically and launching backup tasks. It decreases the execution time of MapReduce jobs in heterogeneous environment. This algorithm can be further improved by focusing on how to account for data locality. In [**4**] authors focuses on techniques which can detect a failed worker quickly. As the size of data stored in a cluster increases, there are more chances of occurrence of failures. Ignoring these failures or delay in detecting the failures may result in high completion time of jobs. To cope up with this problem, two mechanisms are presented- Adaptive interval and reputation based detector to detect the failure node in less time. The adaptive interval technique dynamically configure the expiration time of a job while reputation based detector find out the reputation of the worker. Combining these techniques would give better results as adaptive interval technique is suitable for small jobs and reputation based detector technique is stable for large jobs only. In [**5**] a MapReduce scheduling technique is developed to enhance map task's data locality. This technique is integrated into hadoop default FIFO scheduler and fair scheduler. This matchmaking technique gives results that lead to highest data locality rate and lowest response time for map tasks. The main idea behind this technique is to assign tasks to a node, where local map tasks are preferred over non local map tasks. It focuses on problem of decreasing data transmission in MapReduce cluster.

A MapReduce task scheduling algorithm for deadline constraints (MTSD) is proposed in [**6**] which permits user to specify deadline for a job within which it must be completed. Under MTSD a node classification algorithm is proposed, which classifies nodes into several level in a cluster. Further a novel data distribution model is illuminated which distributes the data among these nodes based on their capacity level. This algorithm works well in multi job environment and improves the data locality about 57%. But in extreme cases, this scheduling technique can't meet the requirements. If deadlines are not set reasonably work can't be completed before deadline. In [**7**] authors presented a two phase execution engine used to hide delay in data transmission caused by reduce tasks. This delay if not handled properly results in degradation of system performance. To handle this problem, the execution engine will partition the reduce tasks in two phases. In first phase, to run the reduce tasks, nodes are selected by the engine to pre-fetch intermediate results i.e. output of map tasks. In second phase, computing and memory resources are allocated by the selected nodes to reduce tasks and run them. During the execution, the data transmission delay can be hidden.

In [**8**] a context aware scheduler for hadoop (CASH) is suggested which utilizes the cluster heterogeneity and workload mix. By making the scheduler aware of cluster heterogeneity can improve the throughput of the system. Cash algorithm classifies the jobs and nodes as CPU or I/O bound and maps the tasks with different demands to the nodes which can fulfill these demands. The algorithm is implemented and tested on Mimuk simulator. When compared with FIFO scheduler, performance is improved by 20-36%. For small jobs the simulator showed an average improvement which needs to be improved further. Nguyen et al. in [**9**] proposed a hybrid scheduler algorithm based on dynamic priority in order to reduce the response time for variable length jobs. The dynamic priorities can accommodate multiple task length, job size and job waiting time by applying an algorithm named greedy fractional knapsack for job task processor assignment. A reordering of task processor assignment is implemented for job execution to account of data availability and preserving data locality. It improves the average response time by 2.1x faster than hadoop fair scheduler. In future, effects of service level will be evaluated on total workload completion time.

In [**10**] authors proposed a job aware scheduling algorithm that overcomes the problems such as inadequate utilization of computing resources, limited applicability towards heterogeneous cluster, arbitrary scheduling of non local map tasks and negligence of small jobs in scheduling. In addition, they evaluate the performance of the proposed algorithm using MapReduce Word Count benchmark. Their experimental results show that the algorithm increases the resource utilization and reduces the average waiting time compared to existing Matchmaking scheduling algorithm.

III. TECHNICAL BACKGROUND

The two core components of hadoop are, HDFS and MapReduce as illustrated in Fig.1 and are explained below:

Hadoop Distributed File System (HDFS)- It is a distributed file system [**11**] that offers high-throughput access to application data and holds huge sets of data. HDFS exploits a master/slave architecture where master includes a single

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 3, March 2016

NameNode that deals with the file system metadata and one or more slave **DataNodes** that stores the real data. A file in an HDFS namespace is divided into a number of blocks and these blocks are stored in place of DataNodes. The NameNode decides the mapping of blocks to the DataNodes. The DataNodes are concerned with read and write operations of the file system. They also concerns about block formation, deletion and replication based on instruction specified by NameNode [**12**].

Hadoop MapReduce [**13**]**-** It is used for parallel processing of large data sets. The term MapReduce in fact refers to two dissimilar tasks that Hadoop programs carry out. First task to be carried is, **the Map Task:** which takes input data and converts it into a set of data, where individual elements are busted down into tuples (key/value pairs). Second is, **the Reduce Task:** This task takes output from a map task as input and merge those data tuples into a smaller set of tuples.

The MapReduce framework consists of single master **JobTracker** and a slave **TaskTracker** for each cluster-node. The master is accountable for resource management, tracking resource consumption and their availability, scheduling the job component tasks on the slaves, monitoring them and re-executing the aborted tasks [12]. The slaves TaskTracker execute the tasks as directed by the master and provide task-status information to the master periodically.

Working of Hadoop-

Stage 1-

A user can submit a job to the Hadoop (a hadoop job client) for mandatory process by identifying the following items:

- The position of the input and output files in the distributed file system.
- The java classes in the form of jar files containing the execution of map and reduce functions.
- The job design by setting different factors specific to the job.

Stage 2-

Afterwards, the Hadoop job client submits the job (jar/executable) and configuration to the JobTracker which then assumes the duty of distributing the software/configuration to the slaves, scheduling and monitoring tasks, providing status and analytical information to the job-client.

Stage 3-

The TaskTrackers on different nodes accomplish the task as per MapReduce design and output of the reduce function is stored into the output files on the file system [12].

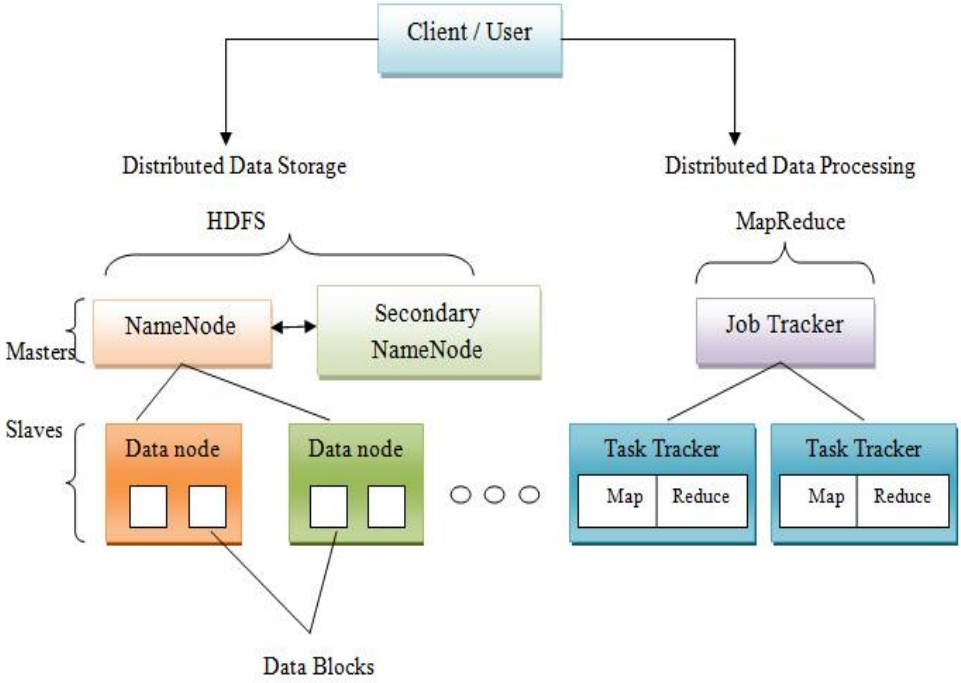

Fig. 1 Components of Hadoop

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 3, March 2016

IV. SCHEDULING IN HADOOP

Various scheduling algorithm has been introduced for execution of huge amount of data. Facebook and yahoo does noteworthy work in developing schedulers i.e. Fair scheduler and Capacity scheduler which are later released to hadoop community. Brief comparison is done between widely used schedulers-

1. Default FIFO Scheduler

The default Hadoop scheduler works using a FIFO (first-in-first-out) queue. After a job is separated into individual tasks, they are weighted down into the queue and allotted to free slots as they become accessible on TaskTracker nodes. Jobs are not accessed based on their priorities. Usually each job would make use of the entire cluster, so jobs had to wait for their turn. However a mutual cluster offers great prospective for contributing large resources to many users, the problem of allocating resources fairly between users requires a better scheduler. Jobs need to finish in a timely manner, while allowing users to get results back in a reasonable time [**14**].

2. Fair Scheduler

The Fair Scheduler [**15**] was developed at Facebook to supervise access to their Hadoop cluster and later released to the Hadoop community. The Fair Scheduler aims to provide every user an equal share of the cluster capacity eventually. Users may assign jobs to pools, with each pool given a certain minimum number of Map and Reduce slots. Free slots in idle pools can be allocated to other pools, while surplus capacity within a pool is shared among jobs. It supports preemption also, so if a pool has not established its equal share for a certain period of time, then the scheduler kills the tasks in pools which are exceeding capacity in order to give the slots to the pool running short of capacity. In addition, administrators may implement priority settings on certain pools. Tasks are therefore scheduled in an interrupted manner, based on priority given within the pool, cluster capacity and usage of their pool. As jobs allocate their tasks to slots in Task Tracker for computation, the scheduler tracks the inconsistency between the amount of time actually used and the ideal allocation for that job. As slots become accessible for scheduling, the next task from the job with the highest time inconsistency is assigned to the next free slot. Over time, this becomes the cause of ensuring that jobs attain roughly equivalent amounts of resources [14]. Shorter jobs are allocated sufficient resources to finish quickly, while longer jobs are assured not to be starved of resources.

3. Capacity Scheduler

Capacity Scheduler [**16**] originally developed at Yahoo deals with a usage scenario where the number of users is large. The Capacity Scheduler allocates various jobs based on number of users to queues with configurable numbers of Map and Reduce slots. Queues that contain jobs are specified with their configured capacity, while free capacity in any queue is shared along with other queues. Within a queue, scheduling works on a modified priority queue with precise user limits, with priorities sync with time a job was submitted, and the priority set allocated to that user and class of job. When a slot in Task Tracker becomes free, the queue with the minimum load is preferred, from which the oldest remaining job is selected. Overall, this has the outcome of enforcing cluster capacity shared among users, rather than among jobs, as was the case in the fair scheduler [14].

Table I- Comparison between scheduling algorithms [17]

Scheduling Algorithm	Taxonomy	Implementation Design	Advantages	Disadvantages
FIFO	Non-Adaptive	schedule jobs based on their priorities as first-come first-out.	cost of whole cluster scheduling process is less. It is easy to implement and efficient.	designed only for single type of job. low performance while running multiple types of jobs. poor response times for short jobs as compared to large jobs.
Fair Scheduling	Adaptive	does an equal distribution of system resources among the jobs.	less complex. works well in both small and large clusters. It offers quick response times for small jobs mixed with larger jobs.	does not consider the job weight for each node.
Capacity	Adaptive	maximize the resource utilization and throughput in multi-node cluster environment.	ensure definite access with the potential to reclaim unused capacity and prioritize jobs within queues over the large cluster.	The most complex among other schedulers.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 3, March 2016

V. CONCLUSION AND FUTURE SCOPE

From the above discussion it is clear that huge amount of data are being stored everyday by various IT companies which in turn are processed by their clients. Every client wants to retrieve data in minimum time. So, it is a challenging task for them to provide best services to their clients. They need to improve performance of the underlying framework periodically which manages all these tasks. In future, concept of pipelining and queue management will be introduced with existing hadoop scheduler. This will reduce the execution time of jobs and utilizes the resources making them available for other job quickly. One of the existing hadoop scheduler will be modified to improve hadoop's performance. The focus will also be on maximum utilization of system resources with complete fairness criteria.

REFERENCES

[1]Dongjin Yoo and Kwang Mong Sim, "A Comparative Review of Job Scheduling for MapReduce," in Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on, Beijing, China, Sep 2011, pp. 353-358.
[2]Jorda Polo et al., "Performance-driven task co-scheduling for MapReduce Environments," in Network Operations and Management Symposium (NOMS), 2010 IEEE, Osaka, Japan, Apr 2010, pp. 373-380.
[3]Quan Chen, Daqiang Zhang, Minyi Guo, Qianni Deng, and Song Guo, "Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment," in Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on, Bradford, England, Jun 2010, pp. 2736 - 2743.
[4]Hao Zhu and Haopeng Chen, "Adaptive failure detection via heartbeat under Hadoop," in Services Computing Conference (APSCC), 2011 IEEE Asia-Pacific, Jeju Island, Dec 2011, pp. 231-238.
[5]Chen He, Ying Lu, and David Swanson, "Matchmaking: A New MapReduce Scheduling Technique," in Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, Athens, Greece, Nov 2011, pp. 40-47.
[6]Zhuo Tang, Junqing Zhou, Kenli Li, and Ruixuan Li, "MTSD: A task scheduling algorithm for MapReduce base on deadline constraints," in Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International, Shanghai, China, May 2012, pp. 2012 - 2018.
[7]Xiaohong Zhang, Guowei Wang, Zijing Yang, and Yang Ding, "A two-phase execution engine of reduce tasks in Hadoop MapReduce," in Systems and Informatics (ICSAI), 2012 IEEE International Conference on, Yantai, China, May 2012, pp. 858 - 864.
[8]K. Arun Kumar, Vamshi Krishna Konishetty, Kaladhar Voruganti, and G. V. Prabhakara Rao, "CASH: Context aware scheduler for Hadoop," in ICACCI '12 Proceedings of the International Conference on Advances in Computing, Communications and Informatics, New York, USA, Aug 2012, pp. 52-61.
[9]Phuong Nguyen, Tyler Simon, Milton Halem, David Chapman, and Quang Le, "A hybrid scheduling algorithm for data intensive workloads in a mapreduce environment," in Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on, Chicago, USA, Nov 2012, pp. 161-167.
[10]Supriya Pati and Mayuri A. Mehta, "Job Aware Scheduling in Hadoop for Heterogeneous Cluster," in Advance Computing Conference (IACC), 2015 IEEE International Conference, Banglore, India, Jun 2015, pp. 778-783.
[11]"https://developer.yahoo.com/hadoop/tutorial/module2.html,".
[12]"http://www.tutorialspoint.com/hadoop/hadoop_introduction.htm,".
[13]"http://www.webopedia.com/TERM/H/hadoop_mapreduce.html,".
[14]B.Thirumala Rao and Dr. L.S.S. Reddy, "Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments," International Journal of Computer Applications(0975-8887), vol. 34, no. 9, pp. 29-33, Nov 2011.
[15]"https://hadoop.apache.org/docs/r1.2.1/fair_scheduler,".
[16]"https://hadoop.apache.org/docs/r1.2.1/capacity_scheduler,".
[17]Seyed Reza Pakize, "A Comprehensive View of Hadoop MapReduce Scheduling," International Journal of Computer Networks and Communications Security, vol. 2, no. 9, pp. 308–317, Sep 2014.

BIOGRAPHY

Anuradha Sharma is currently pursuing M.tech degree in Computer Science and Engineering at Sri Guru Granth Sahib World University, Punjab, India. Her areas of interest in research are cloud computing and big data processing.

Er. Sheenam Malhotra is currently working as Assistant Professor in Department of Computer Science and Engineering at Sri Guru Granth Sahib World University, Punjab, India. Her area of specialization in research is Mobile Agent and Image Processing.