Survey of MapReduce Scheduling Algorithm

Ashwini Khillari[1] ,
Asst. Professor, Department Of IT/ CS
Pillai HOC College of Arts, Science and and Commerce, Rasayani

Darshana Wajekar[2] ,
Asst. Professor, Department Of IT/ CS
Pillai HOC College of Arts, Science and Commerce, Rasayani

Priyanka Sonawane[3]
Asst. Professor, Department Of IT/ CS
Pillai HOC College of Arts, Science and Commerce, Rasayani

Abstract— Recent trends in big data have exposed that the amount of data continues to increase at an exponential rate. This trend has encouraged many researchers over the past few years to survey new research direction of studies related to several areas of big data. Especially, enhancing resources and jobs scheduling are becoming critical since they fundamentally determine even if the applications can accomplish the performance goals in different use cases. Scheduling operate an important task in big data, mainly in compressing the cost of processing and execution time. This paper aims to survey the research undertaken in the field of scheduling in big data platforms.

Index Terms— Big data, Hadoop, Scheduling algorithm

1 INTRODUCTION

Big data is collecting a huge amount of data sets that cannot be processed using traditional computing techniques. In 2005, Mike Cafarella and Doug Cutting provided key by Google and begin a open-source project called HADOOP. Now a days Apache Hadoop is registered trade name of the Apache Software Foundation. Businesses have greatly benefited from data analytics. Companies are analyzing activities such as sales, stock optimization, advertising and consumer support to improve their strategic, tactical business decision, fraud and risk management. Illustrative data-intensive web applications consist of on-line auctions, search engines, customizing content for users, supply-chain management, webmail, on-line retail sales, bio informatics, beyond analytics data, fraud analysis, miscellaneous uses, etc.;

Hadoop is an open-source platform of MapReduce structure, supported by IT companies such as Yahoo, Facebook, Google, IBM, Amazon, YouTube, Linked In to process and analyze their massive amounts of data. It is intended to accelerate from one server machines to hundreds or thousands of machines, each one over local storage and working out, relatively depend on hardware to deliver high possibility are planned to notice and manage failures at the application layer, so deliver a highly available service within the reach of a cluster of computers, which is a level to collapse.

This paper deeds to offer a more widespread meaning of huge information is Big Data that captures its exclusive and character. Dealing with Big information faces "V" challenges. These V-based characterizations represent ten different challenges associated with the main tasks involving big data (as mentioned earlier: capture, cleaning, curation, integration, storage, processing, indexing, search, sharing, transfer, mining, analysis, and visualization).

1. Volume: = lots of data (which I have labeled a "Tonnabytes", to suggest that the actual numerical scale at which the data volume becomes not easy in a particular setting is domain-specific, but we all have the same opinion that we are now dealing with a "ton of bytes").
2. Variety: = complexity, thousands or more features per data item, the curse of dimensionality, combinatorial explosion, many data types, and many data formats.
3. Velocity: = high rate of data and information flowing into and out of our systems, real-time, incoming!
4. Veracity: = necessary and sufficient data to test many different hypotheses, vast training samples for rich micro-scale model-building and model validation, micro-grained "truth" about every object in your data collection, thereby empowering "whole-population analytics".
5. Validity: = data quality, governance, master data management (MDM) on massive, diverse, distributed, heterogeneous, "unclean" data collections.
6. Value: = the all-important V, characterizing the business value, ROI, and potential of big data to transform your organization from top to bottom (including the bottom line).
7. Variability: = dynamic, evolving, spatiotemporal data, time series, seasonal, and any other type of non-static behavior in your data sources, customers, objects of study, etc.
8. Venue: = distributed, heterogeneous data from multiple platforms, from different owners' systems, with different access and formatting requirements, private vs. public cloud.
9. Vocabulary: = schema, data models, semantics, ontologies, taxonomies, and other content- and context-based metadata that describe the data's structure, syntax, content, and provenance.
10. Vagueness: = confusion over the meaning of big data (Is it Hadoop? Is it something that we've always had? What's new about it? What are the tools? Which tools should I use? etc.)

2 Hadoop framework

Hadoop framework consists of two main components:
1. Hadoop's Distributed File System (HDFS)
2. The Map Reduce language

Hadoop's Distributed File System (HDFS)

Given below is the architecture of a Hadoop File System.

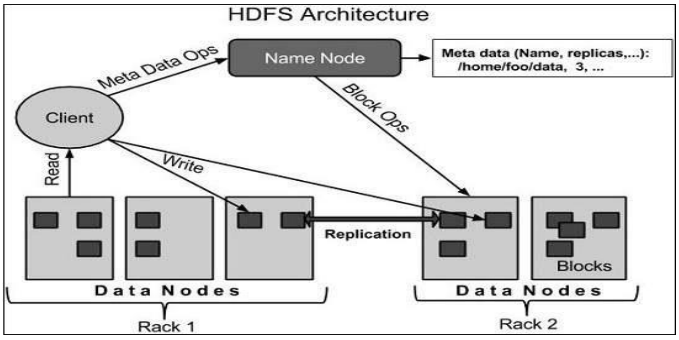

Figure: 1 Hadoop Architecture

HDFS follows the master-slave architecture and it has the following elements.

1. Namenode

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks −

- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

2. Datanode

The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

Block

Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

This paper provided an overview of scheduling in MapReduce and discussed the current scheduling algorithms. The scheduling algorithms are based on strategies, resources, workload, optimization approaches, requirement, and speculative execution. The main objective is to assist researchers to understand the current research improvement and to address future research directions in the area of MapReduce scheduling.

MapReduce

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into *mappers* and *reducers* is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

- **Map stage** − The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
- **Reduce stage** − This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

After completion of the given tasks, the cluster collects and reduces the data to form an appropriate

Terminology

- **PayLoad** − Applications implement the Map and the Reduce functions, and form the core of the job.
- **Mapper** − Mapper maps the input key/value pairs to a set of intermediate key/value pair.
- **NamedNode** − Node that manages the Hadoop Distributed File System (HDFS).
- **DataNode** − Node where data is presented in advance before any processing takes place.
- **MasterNode** − Node where JobTracker runs and which accepts job requests from clients.
- **SlaveNode** − Node where Map and Reduce program

runs.

- **JobTracker** − Schedules jobs and tracks the assign jobs to Task tracker.
- **Task Tracker** − Tracks the task and reports status to JobTracker.
- **Job** − A program is an execution of a Mapper and Reducer across a dataset.
- **Task** − An execution of a Mapper or a Reducer on a slice of data.
- **Task Attempt** − A particular instance of an attempt to execute a task on a SlaveNode.

3 LITERATURE REVIEW

The non-adaptive scheduler is the process of scheduling in which the basic control mechanism is not modified on the basis of the system activity. FIFO [1] is the default Hadoop scheduler and the most popular algorithm in the non-adaptive scheduling of Hadoop MapReduce. Possibly the most straightforward approach to schedule task is to maintain a FIFO run queue based on policies or the solution of some optimization problems.

Delay scheduling [2] is proposed to improve data locality in MapReduce, in a situation where there is a conflict between fairness in scheduling and data locality. The idea is that if the next jobs can be scheduled according to fairness, then that job cannot launch local tasks. Such a solution is suitable if the jobs to be scheduled are less and the task is not been scheduled locally. Thus, delaying its scheduling time can significantly improve data locality. The result shows that delay scheduling can achieve nearly optimal data locality. Moreover, it outperformed fair sharing by its simplicity under a wide variety of scheduling policies. Also, Tan et al. [3] suggested an analytically tractable model under different schedulers such as default FIFO Scheduler and the popular Fair Scheduler for job processing delay distribution in MapReduce.

Casas et al. [4] proposed a genetic algorithm to solve the challenges of scheduling scientific workflows in the cloud computing environment. This is because a scheduler that manages both computing and data-intensive applications is scared. The genetic algorithm crossover and mutation operators were modified to allow the addition/removal of a virtual machine from a given chromosome. The experiment indicated that the proposed genetic algorithm scheduler was found to be better than the state-of-the-art schedulers in terms of makespan and monetary cost.

Capacity scheduler [5] is originally developed at Yahoo to run Hadoop applications as shared, where the fair allocation of computation resource is critical to the user. The main idea is that Hadoop MapReduce cluster is partitioned using the available resources between users who collectively use the cluster based on computing needs. Besides, the capacity scheduler is similar to the fair scheduler, it uses a queue to allocate jobs to users, but with the major difference of using prioritized queue jobs. Each queue is given a configured capacity, which contents a scheduling that operates on a modified priority queue basis with specific user limits. The queue with less number of jobs is selected if the slot becomes available in order to scheduler the task for that job. Generally, this may have an impact on the cluster capacity sharing among users than among jobs, as was the case in the Fair Scheduler.

LATE [6] proposed a new scheduling algorithm called Longest Approximate Time to End (LATE) to improve the performance of the Hadoop scheduler in terms of degradation in the heterogeneous environment. Late has built upon three important concepts prioritizing task to speculate, selecting fast nodes to run on, and capping speculative tasks to prevent thrashing. The idea is based on the assumptions that homogeneous cluster makes progress linearly and has many speculative executions. As the result, many nodes may slow down the overall performance of the Hadoop cluster. The proposed algorithm has improved Hadoop performance by reducing the response time and offer high robust to heterogeneity.

COSHH [7] focused on the application and cluster levels to develop new Hadoop scheduling system named "COSHH." The system considers the heterogeneity of the cluster when executing multiple tasks. The main idea is to offer an improvement in terms of completion time. A typical Hadoop scheduler receives two main messages. from the Hadoop system: a message signaling a new job arrival from a user to store the incoming job in an appropriate queue, and a heartbeat message from a free resource by triggering the routing process to assign a job.

MCP [8] proposed novel speculative execution strategies. The main goal of the scheduler is to choice struggle tasks accurately and then moves them to an available node. This process is to guarantee fairness when the submitted jobs are divided into multiple tasks for particular slots. Unlike Late, the proposed solution uses bandwidth and rate progress in order to choose the slowest task. However, for the remaining time and the progress speed of the tasks are calculated using averages weighted. Moreover, the scheduler can decide on the backup of the task based on the cluster load using a different method such as cost–benefit model, the backups of map tasks is optimized locally, and for each works nodes that are slower is known by the process speed of its map tasks to be executed on the node. The result of the experiment has shown that MCP is 39% faster when running jobs in the cluster and also offer better throughput that is up to 44% compared to default Hadoop.

4. Conclusion

This paper provided an overview of scheduling in MapReduce and discussed the current scheduling algorithms. The scheduling algorithms are based on strategies, resources, workload, optimization approaches, requirement, and speculative execution. The main objective is to assist researchers to understand the current research improvement and to address future research directions in the area of MapReduce scheduling.

REFERENCES

[1]. Hadoop A (2011) Apache Hadoop. https://hadoop.apache.org/. Accessed 3 May 2017

[2]. Zaharia M et al (2010) Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In: Proceedings of the 5th European Conference on Computer Systems. ACM

[3] Tan J, Meng X, Zhang L (2012) Delay tails in MapReduce scheduling. ACM SIGMETRICS Perform Eval Rev 40(1):5–16

[4] Casas I et al (2016) GA-ETI: an enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments. J Comput Sci 26:318–331

[5]. Hadoop A Capacity scheduler guide. https://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html. Accessed 13 June 2017. Zaharia M et al (2008) Improving MapReduce performance in heterogeneous environments. In: OSD

[6] Zaharia M et al (2008) Improving MapReduce performance in heterogeneous environments. In: OSD

[7] Rasooli A, Down DG (2014) COSHH: a classification and optimization based scheduler for heterogeneous Hadoop systems. Future Gener Comput Syst 36:1–15

[8] Qi C, Cheng L, Zhen X (2014) Improving MapReduce performance using smart speculative execution strategy. IEEE Trans Comput 63(4):954–967