Mugla Journal of Science and Technology

# AN OVERVIEW OF HADOOP JOB SCHEDULING ALGORITHMS FOR BIG DATA

Akhtari ZAMEEL, Computer and Information Engineering , Sakarya University, Turkey, akhtari.zameel@ogr.sakarya.edu.tr
(https://orcid.org/0000-0001-7215-0559)
Ahmet ZENGIN, Computer and Information engineering, Sakarya University, Turkey,
azengin@sakarya.edu.tr
(https://orcid.org/ 0000-0003-0384-4148)

Received: 02.06.2022, Accepted: 11.10.2022 Review Article
*Corresponding author DOI: 10.22531/muglajsci.1124422

**Abstract**

*Rapid advancements in Big data systems have occurred over the last several decades. The significant element for attaining high performance is "Job Scheduling" in Big data systems which requires more utmost attention to resolve some challenges of scheduling. To obtain higher performance when processing the big data, proper scheduling is required. Apache Hadoop is most commonly used to manage immense data volumes in an efficient way and also proficient in handling the issues associated with job scheduling. To improve performance of big data systems, we significantly analyzed various Hadoop job scheduling algorithms. To get an overall idea about the scheduling algorithm, this paper presents a rigorous background. This paper made an overview on the fundamental architecture of Hadoop Big data framework, job scheduling and its issues, then reviewed and compared the most important and fundamental Hadoop job scheduling algorithms. In addition, this paper includes a review of other improved algorithms. The primary objective is to present an overview of various scheduling algorithms to improve performance when analyzing big data. This study will also provide appropriate direction in terms of job scheduling algorithm to the researcher according to which characteristics are most significant.*
**Keywords: Big data, hadoop, job scheduling, scheduling algorithms.**

# BÜYÜK VERİLER İÇİN HADOOP İŞ ÇİZELGELEME ALGORİTMALARINA GENEL BAKIŞ

**Özet**

*Büyük veri sistemlerindeki hızlı gelişmeler son on yılda meydana gelmektedir. Büyük veri sistemlerinde yüksek performans elde etmek için en önemli unsur "iş Çizelgeleme"dir. Çizelgelemenin bazı zorluklarını çözmek için daha fazla dikkat gerekmektedir. Büyük verileri işlerken daha yüksek performans elde etmek için uygun çizelgeleme gereklidir. Apache Hadoop, en yaygın olarak çok büyük veri hacimlerini verimli bir şekilde yönetmek için kullanılır ve ayrıca iş çizelgeleme ile ilgili sorunları ele almada yetkindir. Büyük veri sistemlerinin performansını iyileştirmek için çeşitli Hadoop iş çizelgeleme algoritmalarını önemli ölçüde analiz ettik. Çizelgeleme algoritması hakkında genel bir fikir edinmek için bu makale iyi bir arka plan sunmaktadır. Bu makale Hadoop büyük veri çerçevesinin temel mimarisi, iş çizelgeleme ve sorunları hakkında genel bir perspektif sunmaktadır. Ardından en önemli ve temel Hadoop iş çizelgeleme algoritmalarını incelemekte ve karşılaştırmaktadır. Ek olarak makale diğer geliştirilmiş algoritmalarının bir incelemesini sunmaktadır. İlk amacı büyük veriler analiz ederken performansı arttırmak için çeşitli çizelgeleme algoritmalarına genel bir bakış sunmaktır. Bu çalışma aynı zaman da araştırmacıya ihtiyaçlarına göre iş çizelgeleme algoritmasına uygun yönlendirme sağlamaktadır.*
**Anahtar Kelimeler: Büyük veri, hadoop, iş çizelgeleme, çizelgeleme algoritmaları.**

**Cite**

**Zameel, A., Zengin, A., (2022). "An Overview of Hadoop Job Scheduling Algorithms for Big Data", Mugla Journal of Science and Technology, 8(2), 38-48.**

## 1. Introduction

A great revolution in data has seen the light of day in recent years. With the evolution of the Internet which is rapidly developing into Internet of Things (IoT), the idea of "Big Data" emerged [1]. For quality of life, the advantages of IoT devices are enormous in aspects of significant improvement. Simultaneously, it introduces the challenge of dealing with outrageous measures of information. A gigantic volume of information has been produced in recent years as a result of technological innovations from numerous sources. These data sources like the IT sector, data in the form of media streams, transactional information from enterprise applications, electronic gadgets, files, and many more [2].

The characteristics of big data are known as "V" challenges, later these "V" challenges were expanded to include a large number of V's: Velocity, Variety, Volume, Veracity, Variability, Value, Verification, Vulnerability [3, 4]. The management of these highly characterized data is

arduous for traditional database systems. Doug Cutting, a researcher and proprietor of Apache Lucene, designed the Hadoop system to handle these complex big data. Hadoop is very popular among other available big data systems. It is an open-source framework capable of addressing numerous challenges such as high volume of data storage, analysis of data, transfer, capture and data collection, and vulnerability means security and privacy of enormous data [5, 6].

To influence the performance of big data systems, it is important to examine job scheduling. In order to reduce I/O processing time and finest allocation of resources of a job, job scheduling should be considered. Moreover, Job scheduling can anticipate how jobs will use their resources. The scheduling of a job in a computer cluster, on the other hand, is still a well-known outstanding problem. As job scheduling is considered to be an interesting and active field of study in Hadoop, this paper focuses on "Hadoop Job Scheduling" in order to assist researchers in better understanding of Hadoop job scheduling. Working with big data system starts with determining the appropriate algorithm. It is vital to develop an algorithm for determining a node in efforts to advance the Map Reduce process and enhance performance [7]. The ideal algorithm is chosen based on which attributes are most crucial for a certain application. The researcher must carefully choose the most appropriate algorithm in order to improve performance. Goal of this research article is to offer implementers an indication of an ideal scheduling algorithm for their research according to their needs.

To achieve a general understanding of Job Scheduling the following is how the rest of the paper is organized. The structure of this survey study is such that after introduction it will concisely outline an effective big data framework with regard to the overall architecture. Then it will provide information about job scheduling in Hadoop and its issues in chapter 3. In section 4, a survey and comparison on available fundamental job scheduling algorithms with its advantages and disadvantages and also other improved mostly mentioned algorithms has been discussed. Finally, section 5 will represent the conclusion.

## 2. Generalized Features and Architecture of Hadoop System

Various technologies are available for processing and analyzing datasets, however Apache Hadoop is the most commonly utilized and popular [8]. For distributed computing, data storage, and processing, Hadoop is a scalable, accurate and dependable open source platform. It allows to compose, organize and implement applications which was developed in Java at first [9]. When developing distributed applications, Hadoop users should not be concerned with the low-level details of the distributed system, but rather with their business requirements.

### 2.1. Features

For computing communications, Hadoop is adopted by many conventional business and internet enterprises because of its highly scalable features. Paper [10] presented that in 2016 Facebook handled 4 petabytes of data on daily basis and each day in 2017, 2:5 quintillion bytes of information were generated, according to IBM. Amazon uses ad hoc groups to manage enormous volumes of data on a regular basis, according to paper [11, 12, 13].

A substantial dataset is broken down into little segments and spread among multiple nodes, each with its distinct computing and caching capabilities. It delivers a high fault tolerance capability, which implies that in the event of a machine failure, a backup will consistently be present to ensure that processing is not disrupted. As an outcome of this feature, data is more readily available. Because of its extensive data availability, Hadoop is rarely susceptible to failures [14].

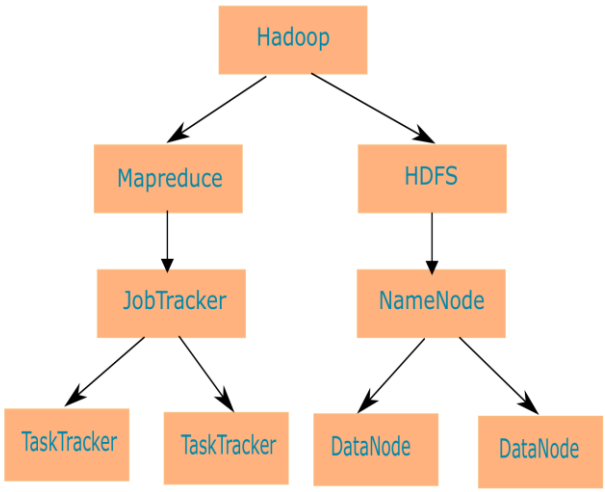


Figure 1. Generalized architecture of Hadoop system.

Hadoop provides the ability to install additional nodes to clusters without requiring them to be reorganized. Hadoop also has the capability of combining multiple data sources. Hadoop can conduct map and reduce operations well regardless of the data type.

### 2.2. Hadoop Architecture

The generalized architecture of the Hadoop system is depicted in Figure 1. Hadoop framework features the Hadoop Core module, which serves as the framework's foundation and encompasses with the operating system's and file system's services and processes. It's a collection of libraries and functions that are shared by Hadoop modules. Another core module of Hadoop is the MapReduce part. This module is responsible for concurrent processing of big data sets and writing applications. It operates with data stored in the Hadoop distributed file system (HDFS) module.

### 2.2.1. MapReduce

Hadoop includes MapReduce, It is a data processing technology developed to reliably process vast datasets in a distributed and parallel fashion using large clusters. The map and reduce functions are the two user-defined functions in a Hadoop MapReduce job [7, 15, 16]. The Mapper function usually splits the data into distinct smaller parts and maps the entire input file.

The map function accepts and returns key(k)/value(v) pairs as input and output. The reducer function will receive the intermediate outcome of the map function to generate the final output. The HDFS, which is parallelized beneath Mapreduce, stores the final output. The whole data processing technology of Map/Reduce follows five phases which is illustrated in Figure 2. Hadoop is in charge of handling the technicalities of important duties, such as task completion verification, task issuance, and replicating information between the cluster's nodes [17].

### 2.2.2. JobTracker

The JobTracker is a component of the MapReduce module which supervises the MapReduce job and functions as a master node. By scheduling system operations to execute on the Task Tracker, JobTracker synchronizes all of the jobs while also maintaining the entire Hadoop cluster's processing resources [18]. It requests NameNode to identify the data in HDFS that necessitates to be executed. To evaluate advancement of the task throughout process execution, the Job Tracker can acquire progress reports from the Task Tracker.

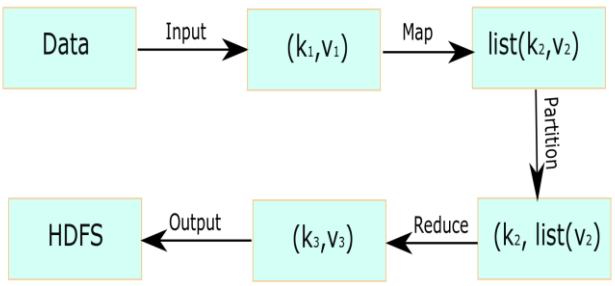


Figure 2. Data Processing of MapReduce Module

### 2.2.3. TaskTracker

The TaskTracker is another component of MapReduce module which receives requests from master node JobTracker and processes the MapReduce job and functions as a slave node. The task tracker's responsibility is to perform the task, and if it fails, it notifies the Job Tracker about the failure [19, 20, 21]. The TaskTracker allocates processing slots for every slave node. The total number of map-reduce slots indicated by the TaskTracker determines the quantity of map-reduce jobs that can be simultaneously performed [7, 22].

### 2.2.4. Hadoop Distributed File System (HDFS)

HDFS (Hadoop Distributed File System) is a file system that is designed to consistently and efficiently store huge quantities of information. It is based on Google File System (GFS), and it enables HDFS to store enormous amounts of data, including meta data and real data, across several devices. A master/slave structure exists in an HDFS cluster [23], name node acting as the master and several data nodes serving the master node. These attributes facilitate data reliability and processing speed. Data is sent via HTTP, which allows access to all information via an internet browser. In the event of failure whether by human or system, system operators can revert to a former version of HDFS following an upgrade.

### 2.2.5. NameNode

HDFS has a single NameNode which is the metadata server. NameNode is liable to map data blocks to DataNodes and manage the file system by indexing all the files of the system. While file-related operations such as locate, transfer, open, delete and rename take place, Name Node handles these requests. NameNode divides the data into numerous pieces and assigns each to a separate Data Node based on its function. It's the only point of failure in the cluster. In the event of NameNode failure, an assistant node called secondary NameNode is available to write checkpoint on HDFS.

### 2.2.6. DataNode

The slave of HDFS is DataNode and responsible for storing real data. HDFS consists of multiple data nodes which enable storing data replication in different locations. As a result, if one DataNode fails, the cluster remains unaffected. In every three seconds, DataNode sends a signal to Name Node to inform its status in the form of heartbeat message [19]. If a DataNode crushes, the NameNode updates its table of list and suspends that specific Data Node from performing any further activities. Read and write operation of HDFS is handled by DataNode. It can also perform construct, replicate, and delete blocks operations dependent on guidance indicated by NameNode [4].

## 3. Job Scheduling

Usually, Job scheduling is the main focus within the analysis of Hadoop since it is a vital interface in influencing the platform's performance. The primary objective of scheduling in Big Data processing is to anticipate the execution and accomplishment of as many activities as feasible with the aid of dealing by efficiently managing and transforming data with the minimal quantity of potential modifications. Another objective of job scheduling is to figure out a way to assign various tasks to appropriate task machines in order to reduce job completion time while increasing task machine usage.

Job scheduling is the process of allocating computational resources from multiple TaskTracker nodes to tasks in a cluster using various methodologies. The working procedure of Hadoop job scheduling is outlined in Figure

3. Clients submit jobs to the job tracker, which are then scheduled by the JobTracker. Then it will send out notifications and assign tasks to task trackers. The TaskTracker requests a new task from the JobTracker. The TaskScheduler delivers the TaskList of undone jobs to the JobTracker once a request for new task is made by the TaskTracker. This TaskTracker is comprises of map and reduce tasks. The JobTracker maintains track of the available map and reduce slots of the TaskTracker for appropriately allocating jobs to them. Afterwards, TaskTracker launches the allocated task after mapping and reducing it [24].

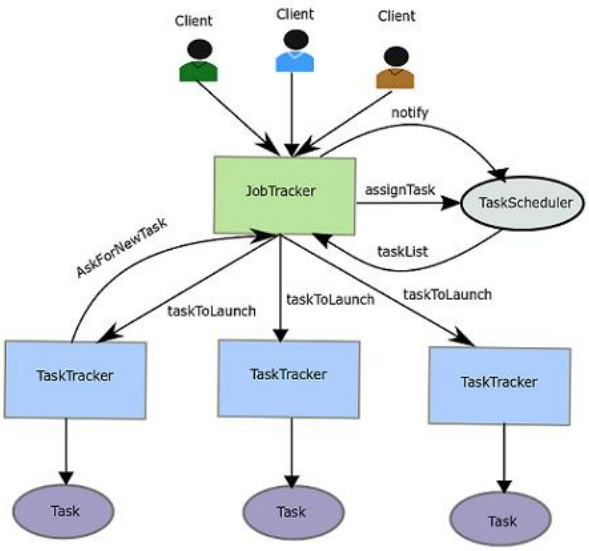


Figure 3. Hadoop Job Scheduling Process

### 3.1. Challenges of Scheduling Algorithm

For the administration of massive data on diverse nodes in Hadoop clusters, optimal scheduling strategies are required. Hadoop schedulers are intended to maximize resource consumption while also improving performance [25]. Certain factors have an impact on the scheduler's overall performance. Therefore, some important challenges must be addressed in order to efficiently construct schedule algorithms. Those are Fairness, Data locality, Availability, Resource Utilization, Throughput and Synchronization [14, 7, 26, 27].

#### 3.1.1. Fairness

The scheduling algorithm's fairness is a metric for how evenly resources are distributed among Map Reduce jobs according to priority levels, volume, and completion time. Certain intensive processes may consume all of the cluster's processing capacity, prompting other light weight tasks to suffer for long periods of time. Avoiding such starvation is the goal of an effective scheduling algorithm. Among the Map and Reduce cycles, Fairness interacts with dependency and locality [28]. In the event that fairness is not ensured, response time and throughput will suffer.

#### 3.1.2. Data Locality

Locality refers to the proximity among an input node and a computation node. The closer the computational node is to the input node, the shorter the data transfer time, which increases throughput. During preparation and execution phases, a k-means method provided by paper [29] that works by putting the first most similar data sets in the single data center. For the betterment of the "stagger Machine" problem, a tasks replication technique was utilized in the MapReduce framework, which repeats tasks replication until all tasks are accomplished [30].

#### 3.1.3. Availability

The term availability refers to the mean time among failures and repetitions of user request. Another definition of availability is the period during which the system facilitates and manages to run continuously in order to fulfill new requests. Following the verification of the scheduler, the overall tasks must be accomplished completely.

#### 3.1.4. Resource Utilization

The utilization of resources is a key feature in numerous scientific fields. The significance arises from a scarcity of resources in comparison to the massive quantity of data and processing power required. The efficiency with which a scheduling algorithm allocates resources between processes is measured by resource utilization. Parallel to the communication utilization, the scheduler should make good use of the available resources, in particular the local computing device hardware. In order to concentrate a number of tasks in clusters and achieve good results, it is vital to optimize and manage these resources.

#### 3.1.5. Synchronization

The impact of synchronization in MapReduce cannot be overstated. The technique of sending intermediate outputs from Map processes as inputs to Reduce processes is characterized as synchronization [4, 7, 31]. The "shuffle and sort" approach is used to establish synchronization. In heterogeneous systems, the issue is exacerbated by the fact that each node's capabilities and configuration differ. As a result, Map job execution times vary, which has an impact on the Reduce phase. The reduce operation will not be able to begin unless all mappers have completed their work. It accelerates in limiting within the overall performance of the Hadoop cluster.

#### 3.1.6. Throughput

In addition to data locality and utilization, good scheduling algorithms should provide high throughput. The number of jobs to be completed in a certain amount of time is known as throughput. A scheduling algorithm is designed to maximize the system's overall throughput.

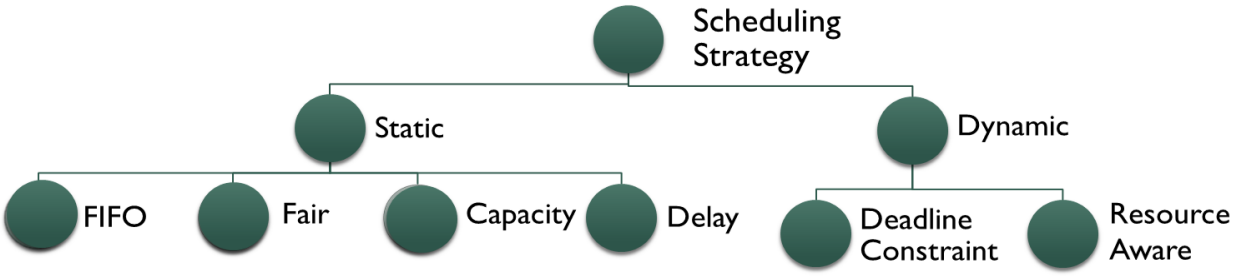


Figure 4. Classification of Scheduling Algorithm

## 4. Overview of Job Scheduling Algorithms

In the character of Hadoop, jobs must share the cluster resources, so a scheduling strategy is required to determine when and where a job will be executed. By effectively assigning tasks to the processors, scheduling aims to reduce completion time, enhance throughput, reduce overhead, and equalize available resources in a parallel program. Job scheduling greatly enhances the performance of a Hadoop cluster according to Jyoti V Gautam's study [32].

### 4.1. Classification

Many scheduling algorithms have been proposed and developed by various researchers which can be classified based on various parameters. To better comprehend these algorithms, classification of Hadoop scheduling algorithm is summarized in Figure 4. Scheduling algorithms can be classified into Static and Dynamic categories based on their scheduling strategy.

Static [33] or offline scheduling involves assigning jobs to processors prior to execution of a program, and information about task execution time and computing resources is available during compilation time. The purpose of static scheduling is to reduce the amount of time required for present programs to run. The assignment of jobs to processors in dynamic or online scheduling takes place during execution time, with little prior information of a job's resource requirements, and even the job environment is uncertain. With distributed computing, dynamic scheduling of jobs necessitates even scheduling of resources throughout multiple geographical regions [34, 35].

In the following section overview of some common and default Static and Dynamic Hadoop scheduler algorithms is provided.

#### 4.1.1. First In First Out (FIFO) Scheduling

The inbuilt Hadoop default scheduler is First in First out (FIFO) [36, 37] algorithm that can be employed in a homogenous environment. According to FIFO queue policy, jobs that join the ready queue first get first precedence regardless of the priority and size of the task. When the order in which tasks are completed is unimportant, the FIFO algorithm is applied. Between all schedulers, the FIFO scheduling strategy is the simplest and most efficient [7, 38]. The main disadvantage of this policy is that a lengthy process will induce all jobs to be delayed. Although each and every job must be completed on time, and each job must have a faster response time [39]. The stability of allocating resources among lengthy and small jobs is not considered by FIFO scheduling. As a result, this approach minimizes data locality and leads to job starvation [7].

#### 4.1.2. Fair Scheduler

With the idea of sharing resources equally over time Facebook developed Fair scheduler [40]. Pools are sets of jobs that are created depending on configurable attributes such as user name. Every user has their own pool, with a minimum share allotted to them. The MapReduce task slot, by default, distributes resources fairly among the pools. If a pool's idle slots are not used, the other pools will employ them. If the identical user or pool sends an excessive number of jobs, the fair scheduler can constrain these jobs by identifying them as not executable. The Fair Scheduler supports preemption [41], which means that if a pool has not gained its fair portion for a predetermined period of time, the scheduler module will refuse tasks in pools that are over capacity in order to allocate slots to pools that are under capacity. The main drawback of FIFO is solved by the fair scheduler where smaller jobs are managed more quickly than larger jobs. Another advantage of the fair scheduler is that it allows the system to specify the number of jobs from each pool and user that can be executed concurrently. The problem of fair scheduler is that it does not take into account the weight of each job, resulting in imbalance performance in each pool.

#### 4.1.3. Capacity Scheduler

Capacity Scheduler, which was originally developed at Yahoo [40, 42], addresses a usage platform in which a vast number of users is present, a requirement to ensure a fair allotment of computation resources and scheduling of task allocation between clients.

Table 1. Comparison of Hadoop scheduling algorithms used in Big Data

| Scheduling Algorithm | Key Techniques | Fair Sharing of Resources | Environment | | Priority in Job Queue | Working Cluster | Merit | Demerit |
|---|---|---|---|---|---|---|---|---|
| | | | Homogeneous | Heterogeneous | | | | |
| **FIFO** | Job allocation: Static, Non-adaptive, Non primitive, Execution time | No | ✓ | × | No | Small clusters | Easy to implement, Cost of cluster scheduling is minimal | Starvation, Bad data locality, Small job will wait long time |
| **Fair Scheduling** | Job allocation: Static, Adaptive, Primitive, Execution time | Yes | ✓ | × | Yes | Small clusters | Distributes resources fairly, Smaller jobs are managed quickly than larger jobs | Imbalance performance due to absence of job weight, Complex configuration |
| **Capacity** | Job allocation: Static, Adaptive, Non primitive, Execution time | Yes | ✓ | × | No | Large clusters | Unused capacity of jobs in queues is reused, Increases throughput | Challenging to select appropriate queues |
| **Delay** | Job allocation: Static, Adaptive, Primitive, Data locality | Less than Fair | ✓ | × | Yes | Small clusters | Scheduling is simple, For complex calculations has no overhead | Limited slot for each node, Ineffective for larger queue of jobs |
| **Deadline Constraint** | Job allocation: Dynamic, Adaptive, Primitive, Deadline | Yes | ✓ | ✓ | Yes | Large clusters | Maximize system utilization, Optimize Hadoop implementation | Nodes be uniform in composition leads to increasing cost |
| **Resource Aware** | Job allocation: Dynamic, Adaptive, Primitive, Resource utilization | Yes | ✓ | ✓ | Yes | Large clusters | Better resource utilization in cluster, Job management performance is enhanced | Non preemption of jobs reduction, Need extra capacities to handle network constraints |

Capacity scheduler employs a queue rather than pools, which is allotted once the resources have been distributed within these queues using a configurable map and reduce slot. Each cluster capacity is distributed between clients as opposed to among tasks, as in the fair scheduler. In case tasks of capacities in lines have additional capacity it kills or pulls down the task. To gain control of the queues, a security technique is designed to guarantee that each client can solely access one of the queues. The capacity scheduler enables job scheduling

depending on priority in terms of time. In a cluster environment, capacity scheduling greatly increases resource utilization and throughput. It also ensures that the unused capacity of jobs in queues is reused. With capacity scheduling it is challenging to select appropriate queues. The issue of capacity scheduling is with context to awaiting jobs, and it has some drawbacks in ensuring the cluster's stability and fairness from a queue and single user [7].

#### 4.1.4. Delay Scheduler

To address the issue of fair scheduling, Facebook created delay scheduling, which attempts to improve locality through the application of a waiting strategy. This scheduling is added by means of enhancing Map Reduce with data locality to improve its performance and the quickest response time for the Map assignment [43]. The reallocation of resources is accomplished by eradicating the pending tasks of preceding tasks in order to manage a slot for that task and waiting for a task to be accomplished in the assigned slots for the latest task [3]. If data for a task is not present, the task tracker will wait for a predetermined amount of time. The scheduler examines at the job's size to determine if there's a request for task allocation from a local node, and if there is, it skips the job which is excessively small and looks for an available subsequent job to execute. If job skipping persists over an extended period of time, a nonlocal task is created in order to avoid starvation. This scheduling is simple and for complex calculations, has no overhead. Disadvantage of delay scheduling is that it has a limited slot for each node. When most of the jobs are substantially more than an average job, the delay scheduling strategy is ineffective.

#### 4.1.5. Deadline Constraint Scheduler

In Hadoop clusters, deadlines are prioritized by anticipating job completion durations [44]. Then assigning them to slots which can be able to process them within a set time limit, which is given by the user. By simply disregarding new work that cannot be handled within their timeframe, this algorithm guarantees that jobs with an appropriate deadline are scheduled for completion. This is obtained with the aid of estimating the due date and comparing it to the time it takes to complete the task [45]. In case the schedulability assessment fails, the user will be prompted to set a new due date, and if that is successful, the task will be rescheduled. It emphasizes on challenges such as meeting deadlines and maximizing system utilization [4, 7, 42]. There is a penalty associated with the requirement that the nodes be uniform in composition.

#### 4.1.6. Resource Aware Scheduler

Scheduling regarding resource use [46] and minimizing resource demand on systems is an important field of study in Hadoop. Currently, the system contains a variety of heterogeneous nodes that provide node assignment variation. Prior schedulers did not take into account the resource availability in greater depth and allotted resources of a specified size [4, 47]. This strategy attempts to reduce resource dissipation while increasing utilization. These schemes cognizance on how efficaciously useful resource usage has been carried out with diverse forms of usage like IO, memory, disk, network and CPU utilization [48, 49] . Two distinct techniques to Resource Aware Scheduling are "Free Slot Filtering" and "Dynamic Free Slot Advertisement." Job management performance is enhanced with a resource aware scheduler. The disadvantages of this scheduling are that it does not support the preemption of reduction jobs and that extra capacities to handle network constraints are required.

### 4.2. Comparison of Scheduling Algorithms

The principal purpose of this research article is to investigate and analyze various types scheduling algorithms and strategies that can be used to improve Hadoop operating efficiency when dealing with enormous datasets. The fact that different jobs have distinctive quality specifications is a vital open area of concern in scheduling algorithms. Many jobs necessitates equilibrium with the other specifications. To better comprehend these algorithms this comparison is based on some conspicuous Hadoop scheduler properties. Such as key techniques, job allocation, area, fairness in resources sharing, environment, priority in job queue, working cluster and their merits and demerits. These criterias will offer the implementers an indication of an ideal scheduling algorithm for their research. Comparison of different scheduling algorithms used in big data is presented in Table 1.

From Table 1, Fair and Capacity Scheduler were found to be developed for short and production jobs, and overcoming the fairness issue. Due to its ability to deliver quick response times for both small and large jobs mixed together, the fair scheduler is advantageous when there are a variety of jobs present. The capacity scheduler is the best option for managing a big Hadoop cluster with several clients, various task kinds, and varying priorities in order to assure rapid access with the ability to utilize idle capacity and prioritize jobs inside queues. Delay scheduling techniques are developed to solve the locality problem. In a deadline constraint scheduler, the user specifies the work due date, and the job is accomplished in real time. In the field of Big Data processing, a job scheduling algorithm is a crucial area of research.

### 4.3. Other Improved Algorithms

Many experts are studying on ways to improve Hadoop's scheduling policies. An overview of some potential novel scheduling algorithms, in addition some immensely-cited and well-established ones, is offered in these review articles.

In [50], the authors suggested "The Gallery Scheduler," an upgraded version of fair scheduler that enables all tasks to be executed by a default or defined configuration file, restricting the amount of tasks per user and pool. However, the system has to wait for the recently arrived

jobs in the Scheduler queue till several of the currently running tasks are completed before it can move on to them.

When TaskTracker is available, the scheduling capabilities choose the queue with the highest available resources based on the task's priority. In the event that the task requires a considerable amount of memory, this scheduling technique guarantees that there is appropriate free Memory in TaskTracker to perform it. This way, resource demands issues can always be addressed right away according to article [51].

The study in [52] offers an algorithm named LsPS (Leveraging size Patterns Scheduler) for managing bursty workloads in a multi-user context that uses knowledge of workload patterns by dynamically modifying resource shares between scheduling algorithms and users for each user to minimize average task response times. This is accomplished by giving users a slot sharing ratio that is inversely proportionate to their task average sizes. With various users, varied clusters, and heterogeneous workloads, the experimental findings show promising results in enterprise scenarios.

Context aware scheduling was proposed by a researcher in [53], with the principal aim of boosting Hadoop scheduling by allowing for dynamic changes in resource availability. Two essential metrics are used to create the layout. First, a large proportion of Map Reduce processes operate on a regular basis and have similar CPU characteristics, network and disk requirements. Second, nodes in a Hadoop cluster grow heterogeneous with time as newer nodes replace existing ones owing to errors. If Hadoop does not execute task preemption, speculative jobs and a context-aware scheduler could assist to avoid inefficiencies caused by resource unpredictability.

In the course of the reduce phase, Hadoop jobs have unbalanced workloads and inefficiently leverage the available computational and network resources. In this research [54], authors introduce two algorithms for optimizing the Locality-Enhanced Load Balance: the Locality-Based Balanced Schedule (LBBS) and the Overlapping-Based Resource Utilization (OBRU) to overcome these issues. Local reduction, shuffle, and final reduce are the three phases of the process. Both of these algorithms improve load balancing significantly while utilizing eight DataNodes, according to the findings of experiments.

In order to minimize the trade-offs between fairness and data localization while job scheduling, the Dynamic Task Splitting Scheduler (DTSS) is presented [55]. According to evaluation and experiment findings, modifying the task split's proportion can enhance fairness and performance. However, researchers must proceed cautiously while adopting the DTSS scheduler because it is ineffective when jobs have convenient access to data-local nodes.

This paper [56] suggested a new method for scheduling tasks and/or jobs in a Big Data Cluster that focuses on enhancing the NameNode's task distribution to the data nodes. This task scheduler outperforms the existing task schedulers: FIFO Scheduler and Capacity Scheduler, as evidenced by the significant results gained.

Recently the Densest-Job-Set-First (DJSF) approach was proposed in this research work [57] as an enhanced heuristic job scheduling mechanism with the goal of increasing system throughput and lowering the average Job Completion Time (JCT). They define the work packing problem, which incorporates maximizing the job sets' completion efficiency, and deploy the k-means clustering technique to achieve JCT alignment. When the k-means clustering approach yields excessive groups, the number of jobs in each job unit decreases, which is a disadvantage for the job packing method to pick and pack appropriate jobs from a job group [56].

By integrating Round robin and the priority scheduling technique, a hybrid scheduler is presented in [58]. The optimization of this approach involves less context switches. Additionally, it overcame every drawback associated with both Round robin and Priority Schedulers.

After prioritizing the jobs with the aid of the k-means clustering technique, authors in [59] used the MapReduce framework to schedule tasks for big data using Hadoop. Here, the energy optimization is carried out using the FireFly Algorithm (FA) and BAT (FF-BAT) technique. The present resource is chosen in the Hadoop map-phase by combining the firefly algorithm (FFA) with the bat algorithm.

For focusing on the scaling issue of data locality rate with shorter completion times, a localization ID and dynamic priority-based hybrid scheduling algorithm (HybSMRP) was presented in [60]. Dynamic priority assists in choosing which tasks and tasks should be allocated to the resource node that is currently accessible. Each slave node in the cluster is given an ID so that a reasonable amount of data can be executed locally. This algorithm guarantees a high data locality rate by avoiding resource waste.

The conventional Hadoop MapReduce fault tolerance technique leads to significant performance disadvantages for processing tasks during failure recovery. For the betterment of this area, Fast Recovery MapReduce (FAR-MR), a fault recovery technique, is suggested to ensure effective recovery [61]. To facilitate rapid recovery from task failure and node failure, this incorporates a novel fault tolerance method that mixes decentralized checkpointing and proactive push mechanisms.

This research [62] proposed an optimal Simulated Annealing (SA) metaheuristic approach for flight scheduling issues and delays. By employing the constraints utilized to avoid flight delays, this strategy yielded the most ideal results. The evaluation is carried out by comparison with the genetic algorithm (GA) and the artificial bee colony (ABC) algorithm. When compared to the conventional methodology, issue

solving using metaheuristic approaches was better since the researched space's dimensions expanded.

The MapReduce framework was the main emphasis of the authors' study in [13, 14, 42, 63-65], as well as its limitations, problems with job scheduling between nodes, and other algorithms presented by different academics. These algorithms were then categorized based on a variety of performance-related quality indicators in some of these studies. However, these papers are mostly oriented on task scheduling or considering dataset of specific platform, whereas we offered a number of algorithms that address a number of Hadoop scheduler concerns. Additionally, this paper provided extensive information on variety of hybrid job scheduling techniques. Our intention is to give researchers proper guidelines to help them with their particular issues, such as load balancing, high data locality, and minimizing competition time. This study gives the implementer appropriate research direction in terms of selecting Hadoop scheduler.

## 5. Conclusion

Because of the rapid expansion of mobile devices and other gadget sensors associated with the Internet, the volume and varieties of data is growing daily. With the aid of numerous applications in various industries, humans process a substantial volume of data. These applications necessitate extensive input/output processing as well as job scheduling. According to reports, IO processing consumes 69% of the resources and 79% of the time for jobs at the Facebook and Microsoft Bing data center. Therefore, Hadoop is a future-oriented technology. Hadoop is frequently presented as the platform that organization needs to address a range of issues. A fundamental understanding of big data is presented, as well as a well-known big data platform, Hadoop, and its essential components. Job scheduling exemplifies a crucial perspective for achieving big data analysis performance. We delivered an assessment of some of the issues of job scheduling algorithms in big data processing in this article. The employment of various Scheduling algorithms to appropriately employ resources was reviewed. As a result of this comparative investigation, we draw the conclusion that Hadoop's core schedulers are ineffective at dealing with a number of problems, including locality, fairness, and speculative execution.

Numerous enhanced schedulers or strategies have been developed by various researchers to address the aforementioned problems. To reduce the time required for job execution and therefore enhance performance, this area still need further focus. Scholars determine the best algorithm for a specific application according to which characteristics are most significant. Following an examination of several relevant research on Hadoop scheduling, one result that comes out is that the basic Hadoop algorithms for scheduling have scope for development. Furthermore, traditional job scheduling algorithms are either focused on a user-centric or a resource-centric approach, since they are unable to acknowledge both elements at the same time. Significant constraints and issues must be overcome with the aid of the researchers in the future in order to guarantee the long-term viability of data management and job scheduling. In the time ahead, it can be improved by introducing or eliminating a few limitations, or taking into account all of the parameters discussed above and rendering it more adaptable.

This analysis is conducted to give academics an appropriate guideline in determining the propitious region regarding the Hadoop scheduling algorithm. We've observed that the study of Hadoop scheduling is a promising subject of study in which further research is required to improve Hadoop scheduling algorithms. Big data warehouses that serve multiple users or organizations are an example of how Hadoop is developing along with its usage paradigms. In order to make better use of cluster resources for big data analytics, the additional flexibility that Hadoop offers is a significant step forward.

## 6. References

[1] Zameel, A., Najmuldeen, M., and Gormus, S., "Context-Aware Caching in Wireless IoT Networks", *11th International Conference on Electrical and Electronics Engineering (ELECO), IEEE*, 2019, pp. 712-717.

[2] Seethalakshmi, V., Govindasamy, V., & Akila, V., "Job scheduling in big data-a survey", *International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC) IEEE,* 2018, pp. 023-031.

[3] Deshai, N., Venkataramana, S., Hemalatha, I., & Varma, G. P. S., "A Study on Big Data Hadoop Map Reduce Job Scheduling", *International Journal of Engineering & Technology*, *7*(3), 59-65, 2017.

[4] Mohamed, E., & Hong, Z., "Hadoop-MapReduce job scheduling algorithms survey", *7th International Conference on Cloud Computing and Big Data (CCBD)*, *IEEE*, 2016, pp. 237-242.

[5] Singh, D., Reddy, C.K., "A survey on platforms for big data analytics", *Journal of big data*, 2(1): p. 1-20, 2015.

[6] Nagina, D. and Dhingra, S., "Scheduling algorithms in big data: A survey", *Int. J. Eng. Comput. Sci*, 5(8): p. 17737-17743, 2016.

[7] Cheng, D., Zhou, X., Lama, P., Wu, J., & Jiang, C., "Cross-platform resource scheduling for spark and mapreduce on yarn", *IEEE Transactions on Computers*, *66*(8), 1341-1353, 2017.

[8] Apache Hadoop. (2021, November 11) [online]. Available: http://hadoop.apache.org.

[9] Hamad, F. and Alawamrah, A., "Measuring the Performance of Parallel Information Processing in Solving Linear Equation Using Multiprocessor Supercomputer", *Modern Applied Science*, 12(3): p. 74, 2018.

[10] Liu, J., Pacitti, E. and Valduriez, P., "A survey of scheduling frameworks in big data systems", *International Journal of Cloud Computing*, 7(2): p. 103-128, 2018.

[11] Guo, Y., Wu, L., Yu, W., Wu, B., & Wang, X., "The improved job scheduling algorithm of Hadoop platform", *arXiv preprint arXiv :*1506.03004, 2015.
[12] Hudaib, A.A. and Fakhouri, H.N., "An automated approach for software fault detection and recovery", *Communications and Network,* 8(03): p. 158, 2016.
[13] Hamad, F., "An overview of Hadoop scheduler algorithms", *Modern applied science,* 12(8): p. 69, 2018.
[14] Usama, M., Liu, M., and Chen, M., "Job schedulers for Big data processing in Hadoop environment: testing real-life schedulers using benchmark programs", *Digital communications and networks*, *Elsevier*, 2017.
[15] Hannan, S.A., "An overview on big data and hadoop", *International Journal of Computer Applications*, 154(10), 2016.
[16] Dai, X. and Bensaou, B., "Scheduling for response time in Hadoop MapReduce" in 2016 IEEE *International Conference on Communications (ICC) IEEE*, 2016, pp. 1-6.
[17] Shi, Y., Zhang, K., Cui, L., Liu, L., Zheng, Y., Zhang, S., & Yu, H., "MapReduce short jobs optimization based on resource reuse", *Microprocessors and Microsystems*, *47*, 178-187, 2016.
[18] Li, X., Jiang, T., & Ruiz, R., "Heuristics for periodical batch job scheduling in a MapReduce computing framework", *Information Sciences*, 326: p. 119-133, 2016.
[19] Ghazi, M. R., & Gangodkar, D., "Hadoop, MapReduce and HDFS: a developers perspective", *Procedia Computer Science*, 48: p. 45-50, 2015.
[20] Liroz-Gistau, M., Akbarinia, R., Agrawal, D., & Valduriez, P., "FP-Hadoop: Efficient processing of skewed MapReduce jobs", *Information Systems*, *60*, 69-84, 2016.
[21] Singla, M., "A survey on Static and Dynamic Hadoop Schedulers", *Advances in Computational Sciences and Technology*, 10(8): p. 2317-2325, 2017.
[22] Rao, B. T., & Reddy, L. S. S., "Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment", *arXiv preprint arXiv*: 1207.0780, 2012.
[23] Mavridis, I. and Karatza, H., "Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark", *Journal of Systems and Software,* 125: p. 133-151, 2017.
[24] Xue, T., You, X., Yan, M., "Research on Hadoop job scheduling based on an improved genetic algorithm", *INTERNATIONAL JOURNAL OF GRID AND DISTRIBUTED COMPUTING*, 10(2): p. 1-12, 2017.
[25] Suresh, S. and Gopalan, N. P., "An optimal task selection scheme for Hadoop scheduling", *IERI Procedia*, 10: p. 70-75, 2014.
[26] Mana, S.C., "A feature based comparison study of big data scheduling algorithms", 2018 *International Conference on Computer, Communication, and Signal Processing (ICCCSP) IEEE*. 2018.
[27] Al-Sayyed, R. M., Fakhouri, H. N., Murad, S. F., & Fakhouri, S. N., "CACS: Cloud Environment Autonomic Computing System", *Journal of Software Engineering and Applications*, *10*(03), 273, 2017.
[28] Wang, Z. and Shen, Y., "Job-aware scheduling for big data processing", 2015 *International Conference on Cloud Computing and Big Data (CCBD), IEEE*, 2015, pp. 177-180.
[29] Yuan, D., Yang, Y., Liu, X., & Chen, J., "A data placement strategy in scientific cloud workflows" *Future Generation Computer Systems*, *26*(8), 1200-1214, 2010.
[30] Chen, Q., Zhang, D., Guo, M., Deng, Q., & Guo, S., "Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment". In *2010 10th IEEE International Conference on Computer and Information Technology, IEEE,* 2010, pp. 2736-2743.
[31] Jia, Z., Zhou, R., Zhu, C., Wang, L., Gao, W., Shi, Y., Zhan, J. and Zhang, L., "The implications of diverse applications and scalable data sets in benchmarking big data systems", *In Specifying Big Data Benchmarks*, Springer, Berlin, Heidelberg, 2012, pp. 44-59.
[32] Alam, A. and Ahmed, J., "Hadoop architecture and its issues" in 2014 *International Conference on Computational Science and Computational Intelligence, IEEE,* 2014.
[33] Casavant, T. L. and Kuhl, J. G., "A taxonomy of scheduling in general-purpose distributed computing systems". *IEEE Transactions on software engineering,* 14(2): p. 141-154, 1988.
[34] Abraham, A., Buyya, R. and Nath, B., "Nature's heuristics for scheduling jobs on computational grids", *The 8th IEEE international conference on advanced computing and communications (ADCOM 2000)*, 2000.
[35] Senthilkumar, M. and Ilango, P., "A survey on job scheduling in big data", *Cybernetics and Information Technologies,* 16(3): p. 35-51, 2016.
[36] Brahmwar, M., Kumar, M., and Sikka, G., "Tolhit–a scheduling algorithm for hadoop cluster", *Procedia Computer Science,* 89: p. 203-208, 2016.
[37] Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., & Stoica, I., "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling", In *Proceedings of the 5th European conference on Computer systems,* 2010, pp. 265-278.
[38] Divya, S., Kanya Rajesh, R., Rini Mary Nithila, I. and Vinothini, M., "Big Data Analysis and Its Scheduling Policy–Hadoop", *IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN*, 2278-0661, 2015.
[39] Nikhil, B., Riddhikesh, B., & Patil Balu, T. M., "A Survey On Scheduling In Hadoop For Bigdata Processing", *Multidisciplinary Journal of Research in Engineering and Technology*, *2*(3), 497-501, 2015.
[40] Yoo, D. and Sim, K. M., "A comparative review of job scheduling for MapReduce", *2011 IEEE International Conference on Cloud Computing and Intelligence System, IEEE,* 2011.

[41] Patil, A. U., Bagban, T. I., & Pande, A. P., "Recent Job Scheduling Algorithms in Hadoop Cluster Environments: A Survey", *International journal of Advanced Research in computer and communication Engineering,* 4(2), 2015.

[42] Gautam, J. V., Prajapati, H. B., Dabhi, V. K., & Chaudhary, S., "A survey on job scheduling algorithms in big data processing", *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), IEEE*, 2015, pp. 1-11.

[43] Xie, Q., Pundir, M., Lu, Y., Abad, C. L., & Campbell, R. H., "Pandas: robust locality-aware scheduling with stochastic delay optimality", *IEEE/ACM Transactions on Networking*, *25*(2), 662-675, 2016.

[44] Kc, K. and Anyanwu, K., "Scheduling hadoop jobs to meet deadlines", 2010 IEEE Second International Conference on Cloud Computing Technology and Science, IEEE, 2010.

[45] Johannessen, R., Yazidi, A., & Feng, B., "Hadoop MapReduce scheduling paradigms", 2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), IEEE, 2017.

[46] Liu, Z., Zhang, Q., Ahmed, R., Boutaba, R., Liu, Y., & Gong, Z., "Dynamic resource allocation for MapReduce with partitioning skew", *IEEE Transactions on Computers*, *65*(11), 3304-3317, 2016.

[47] Yong, M., Garegrat, N., & Mohan, S., "Towards a resource aware scheduler in hadoop", *Proc. ICWS*, 2009.

[48] Mashayekhy, L., Nejad, M. M., Grosu, D., Lu, D., & Shi, W., "Energy-aware scheduling of mapreduce jobs". In *2014 IEEE International Congress on Big Data,* IEEE, 2014, pp. 32-39.

[49] Khalil, W. A., Torkey, H., & Attiya, G., "Survey of Apache Spark optimized job scheduling in Big Data", *International Journal of Industry and Sustainable Development,* 1(1): p. 39-48, 2020.

[50] Usha, D. and Jenil, A., "A survey of Big Data processing in perspective of Hadoop and MapReduce", *International Journal of Current Engineering and Technology*, 4(2): p. 602-606, 2014.

[51] Dean, J., Ghemawat, S., "MapReduce: a flexible data processing tool", *Communications of the ACM*, 53(1): p. 72-77, 2010.

[52] Yao, Y., Tai, J., Sheng, B., & Mi, N., "A job size-based scheduler for efficient task assignments in Hadoop", *IEEE Trans. Cloud Comput*, 77-83, 2015.

[53] Cassales, G. W., Charão, A. S., Pinheiro, M. K., Souveyet, C., & Steffenel, L. A., "Context-aware scheduling for apache hadoop over pervasive environments", *Procedia Computer Science*, *52*, 202-209, 2015.

[54] Li, J., Wang, J., Lyu, B., Wu, J., & Yang, X., "An improved algorithm for optimizing MapReduce based on locality and overlapping", *Tsinghua Science and Technology*, *23*(6), 744-753, 2018.

[55] Xu, Y., & Cai, W., "Hadoop job scheduling with dynamic task splitting", *International Conference on Cloud Computing Research and Innovation (ICCCRI)* IEEE, 2015, pp. 120-129.

[56] Hadjar, K. and Jedidi, A., "A new approach for scheduling tasks and/or jobs in big data cluster", *2019 4th MEC International Conference on Big Data and Smart City (ICBDSC), IEEE*, 2019.

[57] Hu, Z. and Li, D., "Improved heuristic job scheduling method to enhance throughput for big data analytics", *Tsinghua Science and Technology*, 27(2): p. 344-357, 2021.

[58] Rao, B. T., Susmitha, M., Swathi, T., & Akhil, G. "Implementation Of Hybrid Scheduler In Hadoop". *International Journal of Engineering & Technology,* 7(2.7), 868-871, 2018.

[59] Senthilkumar, M., "Energy-aware task scheduling using hybrid firefly-bat (ffabat) in big data". *Cybern Inf Technol,* 18(2), 98-111, 2018.

[60] Gandomi, A., Reshadi, M., Movaghar, A., & Khademzadeh, A., "HybSMRP: a hybrid scheduling algorithm in Hadoop MapReduce framework". *Journal of Big Data*, 6(1), 1-16, 2019.

[61] Zhu, Y., Samsudin, J., Kanagavelu, R., Zhang, W., Wang, L., Aye, T. T., & Goh, R. S. M., "Fast Recovery MapReduce (FAR-MR) to accelerate failure recovery in big data applications". *The Journal of Supercomputing*, 76(5), 3572-3588, 2020.

[62] Erdem, E., Aydın, T., & Erkayman, B., "Flight scheduling incorporating bad weather conditions through big data analytics: A comparison of metaheuristics". *Expert Systems*, 38(8), e12752, 2021.

[63] Dhulavvagol, P. M., Totad, S. G., & Sourabh, S., "Performance analysis of job scheduling algorithms on Hadoop multi-cluster environment". *In Emerging Research in Electronics, Computer Science and Technology*, Springer, Singapore, 2019, pp. 457-470.

[64] Kalia, K., & Gupta, N., "Analysis of hadoop MapReduce scheduling in heterogeneous environment". *Ain Shams Engineering Journal*, 12(1), 1101-1110, 2021.

[65] Zarei, A., Safari, S., Ahmadi, M., & Mardukhi, F., "Past, Present and Future of Hadoop: A Survey". *arXiv preprint arXiv*:2202.13293, 2022.