Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.elsevier.com/locate/dcan

# Job schedulers for Big data processing in Hadoop environment: testing real-life schedulers using benchmark programs

Mohd Usama, Mengchen Liu, Min Chen [*]

*Embedded and Pervasive Computing(EPIC) Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*

A R T I C L E I N F O

*Keywords:*
Big Data
Hadoop
MapReduce
HDFS
Scheduler
Classification
Locality
Benchmark

A B S T R A C T

At present, big data is very popular, because it has proved to be much successful in many fields such as social media, E-commerce transactions, etc. Big data describes the tools and technologies needed to capture, manage, store, distribute, and analyze petabyte or larger-sized datasets having different structures with high speed. Big data can be structured, unstructured, or semi structured. Hadoop is an open source framework that is used to process large amounts of data in an inexpensive and efficient way, and job scheduling is a key factor for achieving high performance in big data processing. This paper gives an overview of big data and highlights the problems and challenges in big data. It then highlights Hadoop Distributed File System (HDFS), Hadoop MapReduce, and various parameters that affect the performance of job scheduling algorithms in big data such as Job Tracker, Task Tracker, Name Node, Data Node, etc. The primary purpose of this paper is to present a comparative study of job scheduling algorithms along with their experimental results in Hadoop environment. In addition, this paper describes the advantages, disadvantages, features, and drawbacks of various Hadoop job schedulers such as FIFO, Fair, capacity, Deadline Constraints, Delay, LATE, Resource Aware, etc, and provides a comparative study among these schedulers.

## 1. Introduction

### 1.1. Motivation

A significant amount of research has been done in the field of Hadoop Job scheduling; however, there is still a need for research to overcome some of the challenges regarding scheduling of jobs in Hadoop clusters. Industries estimate that 20% of the data is in structure form while the remaining 80% of data is in semi structure form. This is a big challenge not only for volume and variety but also for data processing, which can lead to problems for IO processing and job scheduling. Fig. 1 shows the architecture of a Hadoop distributed system.

As we know, this is an era of Big Data where humans process a significant amount of data, in the range of terabytes or petabytes, using various applications in fields such as science, business, and commerce. Such applications require a considerable amount of input/output processing and spend most of the time in IO processing, which is a major part of job scheduling. It is reported that at the Facebook and Microsoft Bing data center, IO processing requires 79% of the jobs' duration and 69% of the resources. Therefore, here we present a comprehensive study of all Hadoop schedulers, in order to provide implementers an idea on which scheduler is the best fit for which job, so that the execution of a job does not take much time for IO processing and job scheduling becomes much easier. This paper will be useful for both beginners and researchers in understanding Hadoop job scheduling in Big data processing. It will also be useful in developing new ideas for innovations related to Hadoop scheduler.

### 1.2. The definition: big data

Many definitions of big data have been presented by scientists. Among all the definitions, the most popular definition [97] was presented by Doug Laney (2001) in his META Group research note, which describes the characteristics of datasets that cannot be handled by the traditional data management tools described below.

Three V's: volume (size of datasets and storage), velocity (speed of incoming data), and variety (data types). With the development of discussion and increasing research interest in big data, the Three V's have

* Corresponding author.
*E-mail addresses:* usama6832@gmail.com (M. Usama), mengchenliu.cs@qq.com (M. Liu), minchen2012@hust.edu.cn (M. Chen).

https://doi.org/10.1016/j.dcan.2017.07.008
Received 15 June 2017; Accepted 28 July 2017
Available online 26 August 2017

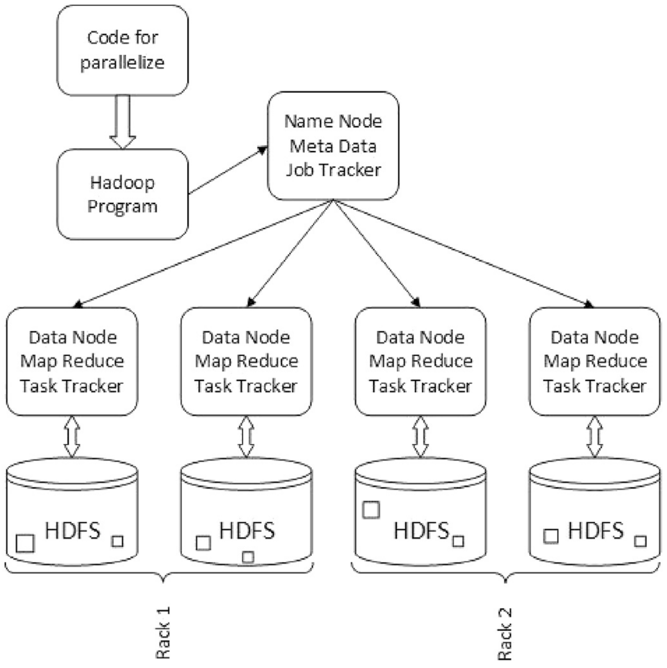

**Fig. 1.** Hadoop distributed system architecture.

been expanded to the Five V's: volume, velocity, variety, veracity (integrity of data), value (usefulness of data), and complexity (degree of interconnection among data structures).

### 1.3. *Characteristics of big data*

Vs of Big data [70]. Velocity of data: velocity means the speed of data processing. The increasing rate at which data flows into an organization has followed a similar pattern to that of volume. Volume of data: Volume refers to the size of data and storage capacity. A large amount of data comes into the storage from many sources such as social media web, business transactions etc. Variety of data: Variety refers to the different types of data. Big data includes a variety of data, both structured and unstructured.

### 1.4. *What is big data?*

Big data [24,32] has come into existence since the last few years. Big data is similar to a conventional database system, but the difference is that it exceeds in size and processing capacity [32]. Big data is very big, moves very fast and includes both structured and unstructured formats of data, whereas conventional databases include only structured format of data [10]. To obtain the benefits of big data, an alternative way must be chosen to process the data [65]. Big data is data that exceeds the processing capacity of conventional database systems [77].

Today's big data [82] plays a vital role in various disciplines and has become popular in computer science and technology, business and finance, banking and online purchasing, oceanography, astronomy, health-care and so on [34]. At present, big data has become a necessity in business analytics and many other fields [2,50]. It provides tremendous benefits for business enterprises [59,60]. Big data is made up of a large number of datasets. The size of these datasets continues to increase day by day as data comes in continuously from different sources such as social media sites, business transactions, personal data, digital photos, etc. Big data has massive amount of unwanted data in both structured and unstructured formats. In structured data, the data is stored in a systematic and well defined manner, while in unstructured data, the data is stored in an unsystematic and undefined manner.

The data coming from Wikipedia, Google, and Facebook have an unstructured format, whereas the data coming from E-commerce transactions have a structured format [3]. Due to the presence of both structured and unstructured data, a number of challenges arise in big data such as data capture, sharing, privacy, data transfer, analysis, storage, search, job scheduling, handling of data, visualization, and fault tolerance [2,57]. It is very complicated and difficult to handle these challenges using traditional database management tools [2].

Traditional data management tools [74] are not capable of processing, analyzing, and scheduling jobs in big data [64]. Therefore, we use a different set of tools to handle these challenges. Hadoop is the most suitable tool to handle all the challenges in big data. Hadoop is an open source software framework for processing big data, and was founded by Apache. It can process large amount of data, in the range of petabytes. Hadoop is a highly reliable cloud computing platform [89], and ensures a high availability of data by making copies of data at different nodes. It is scalable and takes care of the detection and handling of bugs.

There are two components of Hadoop, HDFS and MapReduce. The Hadoop distributed file system is used for data storage, while MapReduce is used for data processing. MapReduce has two functions, Map and Reduce. The functions are both written by the user, and the functions take values as input key value pairs and output the result as a set of key value pairs. First, the Map produces intermediate key value pairs, then the MapReduce function combines all the intermediate values having the same intermediate key as a library, and finally it is passed to the Reduce function. The Reduce function receives the intermediate key with a set of values for that key and merges them to make a smaller set of values.

The aim of scheduling of jobs [84] is to enable faster processing of jobs and to reduce the response time as much as possible by using better techniques for scheduling depending on the jobs, along with the best utilization of resources. FIFO scheduling is default scheduling mode in Hadoop; the jobs coming first get higher priority than those coming later. In some situations, this type of scheduling has a disadvantage, that is, when longer jobs are scheduled prior to shorter jobs, it leads to starvation. Fair scheduling shares the resources equally among all jobs. Capacity scheduling was introduced by Yahoo. It maximizes the utilization of resources and throughput in clusters. LATE scheduling policy was developed to optimize the performance of jobs and to minimize the job response time by detecting slow running processes in a cluster and launching equivalence processes as the background. Facebook uses Delay scheduling, to achieve better performance and lower response time for map tasks by applying changes to MapReduce. In deadline scheduler, the deadline constraints are specified by the user before scheduling the jobs in order to increase system utilization. Resource aware scheduling improves resource utilization; it uses node, master node, and worked node to complete job scheduling. In matchmaking scheduling, each node is marked by the locality marker, which ensures that every node gets an equitable chance to seize a local task. Through this scheduling, high data locality and better cluster utilization is achieved.

There have already been a few review papers on job scheduling algorithms for Big data processing. Yoo. D. et al. (2011) presented a comparative study on job scheduling methods and discussed their strengths and weakness [63]. Rao, B. T. et al. (2012) presented a review on scheduling algorithms and provided guidelines for the improvement of scheduling algorithms in Hadoop MapReduce [77]. Sreedhar C et al. (2015) presented a survey on big data management and discussed various scheduling algorithms in Hadoop. They also discussed the latest advancements related to scheduling algorithms [72]. Jyoti V Gautam et al.(2015) presented a paper on the scheduling policies for Hadoop and performed a comparative study on MapReduce optimization techniques [12].

The remaining parts of the paper are organized as follows. Section II describes challenges for job scheduling in Big Data. Section III describes the architecture, working, features, and requirements of job scheduling in Hadoop. Section IV describes the various Hadoop job schedulers along with their advantages and disadvantages, and compares the various

Hadoop schedulers. Section V evaluates the performance of FIFO, Fair, and Capacity scheduling algorithms based on experimental results. In Section VI, we suggest some directions for future research, and finally the conclusions are provided in Section VII.

## 2. Major challenges for job scheduling in big data

There is a requirement for efficient scheduling algorithms for the management of big data on various nodes in Hadoop clusters. There are various factors that affect the performance of scheduling policies such as data volume (storage), format of data sources (data variety), speed (data velocity), security and privacy, cost, connectivity, and data sharing [4]. To achieve better utilization of resources and management of big data, scheduling policies are designed [5]. The challenges related to job scheduling in big data are summarized as follows:

### 2.1. Data volume (storage)

An important problem of big data is that it is very huge and includes data that is in an unstructured format, which makes it difficult to organize the data for analysis [33,61]. Data includes both structured and unstructured data; the storage of unstructured data is not an easy task [32].

### 2.2. Format of data sources (data variety)

Data comes into big data from various homogeneous as well as heterogeneous resources, and this causes many problems due to the heterogeneity of data resources, data format, and infrastructure [32,33,61].

### 2.3. Speed (data velocity)

Today, everybody expects everything to be done instantaneously. The speed is an important issue in big data [32,61]. Speed of big data is restricted by various problems such as query/retrieval problem, import/export problem, real time/offline problem, and statistical analysis problem [49,53].

### 2.4. Security and privacy

Security is one of the biggest issues for big data [34,52]. When the data is stored, we have to make sure that all the data protection laws are followed. For security and privacy, we need to consider all the data security and privacy rules.

### 2.5. Connectivity and data sharing

Data sharing [96] and connectivity [34] are still issues that need to be considered. At present, a majority of the data points are not yet connected. There are various issues in big data related to connectivity and data sharing such as data standard and interfaces, access permissions, and shared protocols.

### 2.6. Cost

Cost [49] is also an issue in big data. The cost up-gradation issues in big data are cost comparison between master and slave nodes, and upgrade or modification of nodes [52].

## 3. Hadoop framework: architecture, working, features and requirements

Hadoop [25,78,79,81] is an open source software framework for processing big data, which is in the range of petabytes. Doug Cutting, the owner of Apache Lucene, developed Hadoop as a part of his web search engine Apache Nutch. Hadoop is a large scale, batch data processing [46], distributed computing framework [79] for big data storage and analytics [37]. It has the ability to facilitate scalability and takes care of detecting and handling failures. Hadoop ensures high availability of data by creating multiple copies of the data in different locations (nodes) throughout the cluster. There are two main components of Hadoop: HDFS and MapReduce [1].

### 3.1. Hadoop architecture

#### 3.1.1. Hadoop distributed file system (HDFS)

HDFS [13,80,81] is a distributed storage system designed for the storage of distributed files on HDFS nodes [40]. HDFS has a master/slave architecture [40]. An HDFS cluster has a single name node, a master server that manages the file system name-space and controls the access to files by clients [27,36]. The data is divided into 64 MB or 128 MB blocks by the HDFS, and it maintains three copies of each block on distinct locations called Data Nodes [58]. The Name Node executes the file system name-space operations such as opening, closing, and renaming files and directories, while the Data Node is responsible for reserving read and write requests from the file system's clients. It also determines the mapping of blocks to data nodes. The Data nodes perform block creation, deletion, and replication upon instruction from the name node. HDFS gives high throughput access, and is suitable for applications that have large data sets. HDFS is highly fault tolerant and can be deployed on low-cost hardware [28].

#### 3.1.2. Name node

In HDFS file system, the Name Node [81] works as the central part. It is responsible for tracking the clusters, which have the directory tree of all the files in the file system. The data of all these file is not stored by the Name Node. The processes interact with the Name Node whenever they are required to locate a file, or when they need to add, move, or delete any files. The Name Node responds to successful requests by returning a list of relevant Data Node servers where the data is stored. It is the only failure point in an HDFS cluster. HDFS is not considered as a high availability system. It is inaccessible and goes off in cases when the Name Node is down.

#### 3.1.3. Data node

In the HDFS file system, Data Node [81], which is also known as Slave, is responsible for storing the data. There are more than one Data Node in HDFS, and it stores replicated copies of the data in three locations. Both the Data and Name Nodes work together. The Data Node informs the Name Node about all the blocks for which it is responsible, at the time of starting up. There is no effect on the clusters or availability of data when the Data Node goes down. The data is actually stored in the Data Node, which is configured with a large amount of hard disk space.

#### 3.1.4. MapReduce

MapReduce [31,75,81] is a program model for the processing technique based on distributed computing. The MapReduce programming model has two important functions, Map and Reduce [7,21]. The function of Map is to take input data as a set of key value pairs and produce the output as another set of intermediate key value pairs [9]. The Reduce function receives this intermediate key value pair generated by the Map function as an input and reduces this set of intermediate key value pairs into a smaller set of pairs [68]. The Map function is performed after the Reduce function as per the name of the program model, MapReduce [21].

The main benefit of MapReduce model is that it is scalable for data processing over multiple computing nodes [42]. There are three stages in the processing of MapReduce, namely Map, Shuffle, and Reduce [18,22]. The function of Map stage is to process the input data and to create various small packets of data [19]. HDFS is used to store the input data in the form of a file or directory, and these input files are passed to the Map function step by step. In the Reduce stage, the Reduce function receives the output data from the Map stage as its input data, processes it and

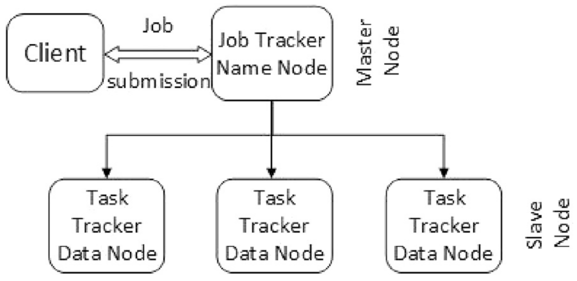

**Fig. 2.** Hadoop mapreduce architecture.

produces a new set of output, which is then stored in the HDFS [29]. Hadoop sends the Map and reduce tasks to the appropriate server in clusters during a MapReduce job [30]. It is the responsibility of Hadoop to manage the details of relevant jobs such as verifying the task completion, issuing tasks, and copying data between the nodes in the cluster [45]. Various computing tasks, which reduce the network traffic, are placed on the nodes with data on local disks. When the tasks are completed, the cluster [88] collects the output data and reduces it to an appropriate form to send it back to the Hadoop server [31,41].

#### *3.1.5. Job tracker*

The Job Tracker [11,81] is a node. The job execution process is controlled by the Job Tracker, and it coordinates all the jobs by scheduling tasks running on the system to run on the Task Tracker [43]. All the processing resources of the Hadoop cluster are maintained by the Job Tracker, and the Job Tracker schedules them to the Task Tracker for execution as per application request. The Job Tracker has the ability to receive status updates from the Task Tracker to track progress during process execution. It performs coordination to handle any failure whenever required. The Job Tracker has to be run on the master node, because it is responsible for coordinating all MapReduce jobs in the cluster.

#### *3.1.6. Task tracker*

Task tracker [43,81] is a node that receives tasks from the Job Tracker. It is the duty of the task tracker to run the task, and if any failure occurs, then it sends a report to the Job Tracker about that failure [41]. The Job Tracker coordinates and helps in resolving the problems that occur during task processing. The processing resources on each slave node are managed by the Task Tracker in the form of processing slots [47]. None of slots for map-reduce tasks are fixed [62]. The number of map-reduce tasks that can be executed at a given time depends on the total number of map-reduce slots indicated by the slave node [77].

### *3.2. Hadoop working*

When a job is submitted to Hadoop, the user needs to specify the location of input as well as output files in HDFS [38]. The Map and Reduce functions are implemented using Java classes in the form of jar files, and different parameters have to be fixed for job execution and job configuration. Once a Hadoop user submits a job and job configuration to the Job Tracker, the Job Tracker distributes the job and configurations to the scheduling tasks and slaves [73]. It also controls the jobs and provides status and diagnosis information to the Hadoop user. The Task Tracker executes tasks on different nodes as per the MapReduce implementation and saves the output in output files on the HDFS. Fig. 2 shows the architecture of Hadoop MapReduce.

Fig. 3. illustrates the traditional method of MapReduce. At present, Hadoop utilizes a new parameter called YARN, which provides a truly new approach to data processing. YARN provides an easy way to handle data processing with a single platform and provides a new perspective to analytics. Fig. 4 shows the details of the YARN architecture.

#### *3.2.1. Hadoop YARN*

In new generation Hadoop frameworks, YARN (Yet Another Resource Negotiator) [71] is a core architectural part (in Fig. 3.), which enables multiple data processing [83] engines such as batch processing, data science, real time streaming, and interactive SQL to handle data processing with a single platform and provides a new perspective to analytics.

The basic purpose of YARN [93] is to split the functionalities of Job Tracker/Task Tracker [19] into separate daemons. In this new version of Hadoop, a global Resource Manager, Slave node manager for each node, Application Master, and container for each application (single job or maybe DAG of jobs) runs on a Node Manager [9].

The Resource Manager and Node Manager [23] work together as a data-computation framework. The global Resource Manager runs as the master daemon that arbitrates cluster resources among the available competing applications in the system. The job of the Resource Manager is to trace the nodes running on the cluster and the resources available on it. It also ensures that the applications submitted by user get the resources whenever required. It is responsible for allocating these resources to the

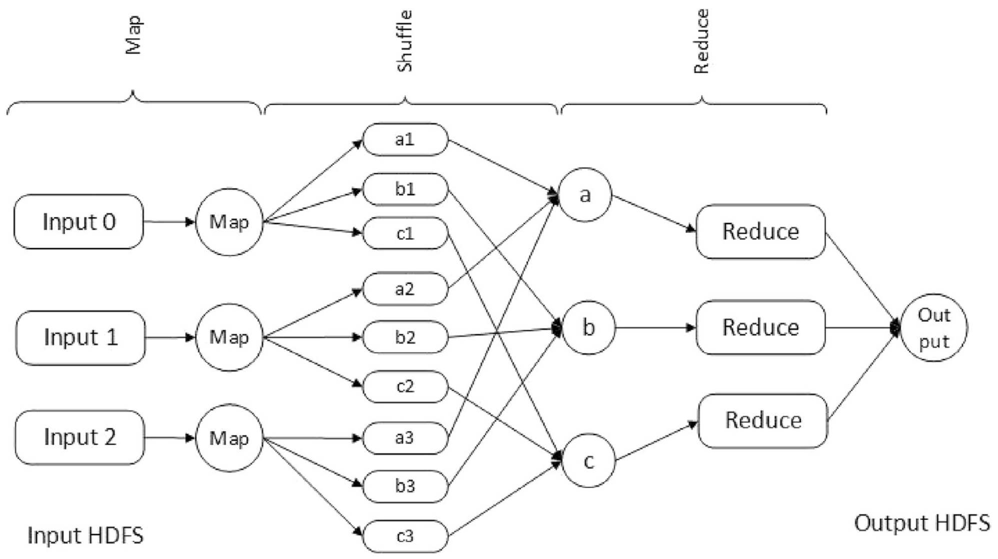

**Fig. 3.** Hadoop map reduce process.

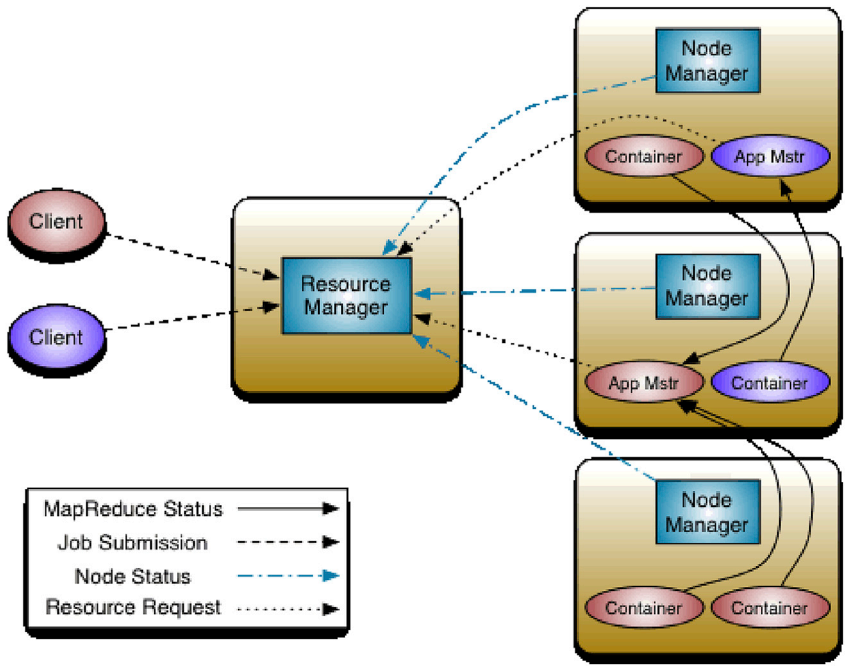

**Fig. 4.** YARN architecture.

user application according to the application priority, queue capacity, ACLs, data locality etc.

In Hadoop 2.0, the Task Tracker is replaced with the Node Manager, which is more generic and efficient [26]. The Node Manager has a number of dynamically created resource containers instead of having a fixed number of Map and Reduce slots as in the Task Tracker. The size of the container depends on the number of resources it contains, such as the memory, disk, CPU, Network IO etc. The Node Manager is responsible for the container's application and it monitors their resource usage such as CPU, disk, memory, network, and reports the same to the Resource Manager.

In Hadoop 2.0, there is a slight change in the design of the Job Tracker, and it is termed as Application Master. The Application Master has the ability to control any kind of task inside the container. Therefore, when a user submits an application, the Application Master coordinates the execution of all tasks within the application. The Application Master also has more responsibilities, which were previously assigned to a single Job Tracker such as monitoring a task, starting failed tasks, calculating total value of application counter and speculatively running slow tasks. Node Manager controls the Application Master and tasks which belong to application running in resource container.

In YARN, the role of distributed application is degraded by MapReduce; however, it is still very popular and useful, and now it is known as MapReduce version 2 (MR2). MR2 is a classical implementation of MapReduce engine that runs top of YARN. MapReduce in Hadoop 2 maintains API compatibility with the previous stable release, and all the functions of MapReduce still work with YARN through a bit of recompiling.

### 3.3. Hadoop features

- It is open source as well as compatible with all platforms as it is Java based.
- New node can be added by Hadoop without changing the cluster and the cluster can adapt more than one node without facing any problem.
- Hadoop performs parallel computing, which makes it very efficient and effective in producing good results.
- It utilizes parallelism of the CPU cores and provides automatic distribution of works and data among machines.
- The Hadoop library itself has been designed in such a way that it has high availability and detects and handles failure at the application layer and does not require the hardware to provide fault tolerance.
- The server can be removed or added dynamically from the cluster and Hadoop operates continuously without any interruption.

### 3.4. Requirements (issues and challenges) regarding job scheduling in hadoop

The Hadoop schedulers are designed for better utilization of resources and performance enhancement [54]. The performance of the scheduler is affected by certain issues, and there is a need for further research efforts to improve the efficiency of Hadoop schedulers. Some of the key research challenges are discussed here, namely Energy efficiency, Mapping scheme, Load balancing, Automation and configuration, data shuffling optimization, Performance optimization, Fairness, Data locality, and Synchronization.

#### 3.4.1. Energy efficiency

In order to support many users and operations involving a large amount of data, there is a continuous growth in the data size in data centers utilizing Hadoop clusters [14]. To realize these operations, a large amount of energy is required in data centers, which increase the overall cost. The minimization of energy in data centers is a big challenge in Hadoop.

#### 3.4.2. Load balancing

There are two stages, Map and Reduce, linked with the partition stage. The partition stage is considered as the main factor for measuring performance. By default, the data is equally portioned by partition algorithms, which handle the system imbalance in case skewed data is encountered. As only the key is considered in processing, and not the data size, load balancing problem [91] occurs.

#### 3.4.3. Mapping scheme

There is a need to assign input sets to the reducer, so that the reducer receives all possible inputs before the execution for each output [55]. No fixed input can be assigned, because the reducer has a limited capacity for accepting inputs, and the individual input size may vary, which leads to an increase in communication cost [56]. Many solutions have been developed to deal with the restriction of input size; however, the optimization of the communication cost between the map and reduce phases has rarely been seen. Therefore, a mapping scheme [90] that could be helpful in the minimization of communication cost is required.

#### 3.4.4. *Automation and configuration*

Automation and configuration [86] helps in the deployment of Hadoop cluster by setting all the parameters. For proper configuration, both the hardware and work amount should be known at the timing of deployment. During configuration, a small mistake could cause inefficient execution of the job, leading to performance degradation. To overcome such types of issues, we need to develop new techniques and algorithms that perform calculations in such a manner that the setting could be done efficiently.

#### 3.4.5. *Optimized data shuffling*

In MapReduce, the overall performance of the system is reduced, because intensive disk input/output increases the execution time during the shuffling phase [6]. Therefore, reducing the execution time during the shuffling phase is more challenging.

#### 3.4.6. *Performance optimization*

Many jobs involve factors such as scheduling, task initialization time, and monitoring of performance degradation ; however, it still does not support features such as overlapping or pipelining of MapReduce phases [8]. In order to improve Map and Reduce, some of the aspects need further optimization such as index creation, fast query execution, and reuse of previous computed results.

#### 3.4.7. *Fairness*

Fairness refer to the fairness in scheduling algorithms. It indicates how fairly of scheduling algorithms divide the resources among users. A fair measure of resources is required among users without starvation. With heavy workload, the entire cluster is used by map-reduce, and because of this, light weight jobs do not get the desired response time. Fairness [92] deals with dependency and locality between the Map and Reduce phases. If there is an equal share of Map and Reduce jobs, and the input files have been distributed in clusters, then there will be a performance degradation in the response time and throughput.

#### 3.4.8. *Data locality*

The distance between the task node and input node is known as locality. The data transfer rate depends on the locality. The data transfer time will be short if the computational node is near the input node. Most of the time, it is not feasible to acquire the node locality, and in such a situation, we acquire rack locality by performing the task at the same rack. The rack is a metal frame used to support the hardware devices.

#### 3.4.9. *Synchronization*

Synchronization [85] is a significant factor in MapReduce. The process of transferring the intermediate output data of the mapping process as the input data to the reduce process is known as synchronization. As the reduce process starts after the completion of the Map process, if any Map process is delayed then the entire process will slow down. A common problem in heterogeneous environments is that, as each node has a unique bandwidth, hardware, and computational facility, it leads to a reduction in the performance of the Hadoop cluster.

## 4. Classification of hadoop job schedulers

The Hadoop job scheduler [44,54] can be classified in terms of the following aspects: environment, priority, resource awareness (such as CPU time, free slot, disk space, I/O utilization), time, and strategies. The main idea behind scheduling is to minimize overhead, resources, and completion time, and to maximize throughput by allocating jobs to the processor [51]. Here, the classification of schedulers is done based on the scheduling strategies [87], time, and resources [39].

### 4.1. *Static scheduler*

Static scheduling strategies allocate a job to the processor before execution of a program begins. The processing resource and job execution time are recognized only at the time of compilation. The main purpose of this kind of schedulers is to minimize the overall processing time of the currently running jobs. FIFO (First in First out), Delay, Capacity, LATE (Longest approximation time to end), and Matchmaking scheduling strategies come under the category of Static scheduling.

### 4.2. *Dynamic scheduler*

Dynamic scheduling [15] strategies allocate a job to the processor at the time of execution of the program. The scheduler has some knowledge about the resource before execution, but the environment in which the job will be executed is totally unknown, and the job will be executed during their life time. In a dynamic environment, the decisions are made and applied to the processor when the execution of the job starts. Resource Aware and Deadline Constrain come under the category of dynamic schedulers.

### 4.3. *Based on resource availability*

Basically, this scheduling strategy is based on the resource requirement of the job. Under this strategy, resource utilization (such as I/O, memory utilization, disk storage, and CPU time), and job performance is improved. Delay scheduler, Matchmaking scheduler, and Resource Aware schedulers are all based on resource availability.

### 4.4. *Time based scheduler*

This scheduling strategy is based on time; here, job completion depends on the user provided deadline. In this scheduling strategy, there is a time limit within which the job must be completed. The user specified deadline then checks whether the job is completed within the given limit or not. Two scheduling strategies, Deadline Constrain and Delay are used for time based job scheduling.

#### 4.4.1. *First In First Out (FIFO) scheduling*

FIFO scheduling [84] policy is the default policy used in Hadoop. This policy gives more preference to the jobs coming in earlier than those coming in later [35]. When new jobs arrive, the Job Tracker pulls the earliest job first from the queue [66]. Here, irrespective of the size of the job or any kind of priority, only the next job is allowed into the queue and the remaining jobs need to wait until the first coming job is executed. FIFO scheduling policy is used when the order of execution of jobs has no importance.

#### 4.4.2. *Advantages*

1. FIFO scheduling technique is the simplest and most efficient among all the schedulers [66]. 2. The jobs are executed in the same order in which they are submitted.

#### 4.4.3. *Disadvantages*

1. A major drawback of FIFO scheduling is that it is not pre-emptive. Therefore, it is not suitable for interactive jobs. 2. Another drawback is that a long-running process will delay all the jobs behind it. 3. FIFO Scheduler does not take into account the balance of resource allocation between long jobs and short jobs. 4. It reduces data locality and starvation of jobs.

#### 4.4.4. *Fair scheduler*

Facebook developed the fair scheduler [63]. The main idea behind fair scheduler is to allocate equal share of resources to each job. It creates groups of jobs based on configurable attributes like user name, called pools. The fair scheduler ensures fairness in sharing of resources between pools. The pools also control job configurable properties, and the configurable properties determine the pool in which a job is placed. All the users have their own pools with a minimum share assigned to each. Minimum share means that a small part of the total number of slots is always achieved by a pool. By default, there is a fair allocation of resources among the pools with the MapReduce task slot. If any pool is free i.e. their are not being used, then their idle slots will be used by the other pools. If the same user or same pool sends too many jobs , then the fair scheduler can limit these jobs by marking the jobs as not runnable. If there is only a single job running at a given time, then it can use the entire cluster.

#### 4.4.5. *Advantages*

1. This scheduler makes a fair and dynamic resource reallocation. 2. It provides faster response to small jobs than large jobs. 3. It has the ability to fix the number of concurrent running jobs from each user and pool.

#### 4.4.6. *Disadvantages*

1. Fair scheduler has more complicated configurations. 2. This scheduler does not consider the weight of each job, which leads to unbalanced performance in each pool/node. 3. Pools have a limitation on the number of running jobs under fair scheduling.

#### 4.4.7. *Capacity scheduler*

The Capacity Scheduler was originally developed by Yahoo [63,84]. The main objective of this scheduler is to maximize the utilization of resources and throughput in a cluster environment. This scheduling algorithm [94] can ensure fair management of computational resources among a large number of users. It uses queues instead of pools unlike fair scheduler, and each queue will be assigned to an organization after the resources have been divided among these queues. In order to get control over the queues, a security mechanism is built to ensure that each organization can access only one of the queues. It can never access the queues of another organization. This scheduler guarantees minimum capacity by having limits on the running tasks and jobs from a single queue. When new jobs arrive in a queue, the resources are assigned back to the previous queue after completion of the currently running jobs. Capacity scheduler allows job scheduling based on priority in an organization's queue.

#### 4.4.8. *Advantages*

1. Capacity scheduling policy maximizes utilization of resources and throughput in cluster environment. 2. This scheduler guarantees the reuse of the unused capacity of the jobs within queues. 3. It also supports the features of hierarchical queues, elasticity, and operability. 4. It can allocate and control memory based on the available hardware resources.

#### 4.4.9. *Disadvantages*

1. Capacity scheduler is the most complex among the three schedulers described above. 2. There is difficulty in choosing proper queues. 3. With regard to pending jobs, it has some limitations in ensuring stability and fairness of the cluster from a queue and single user.

#### 4.4.10. *LATE (longest approximate time to end) scheduler*

The main objective of LATE scheduler [84] is to optimize the performance of jobs and to minimize job response time. When short jobs are running, the response time is quick, which is very important. However, it runs long jobs very slowly due to many issues such as large number of background processes, slow background process, CPU load, unavailability of resource etc. The LATE scheduler finds out process running with slow speed in the cluster, and creates an equivalent process in the background as a backup; such types of processes are called speculative execution of tasks. This scheduler is highly robust for heterogeneity; however, it does not guarantee reliability. By default, this scheduler supports homogeneous clusters.

#### 4.4.11. *Advantages*

1. LATE scheduler optimizes performance of jobs and minimizes job response time as much as possible. 2. This scheduler technique is highly robust in terms of heterogeneity [76].

#### 4.4.12. *Disadvantages*

1. LATE Scheduler does not ensure reliability. 2. This scheduler technique does not guarantee reliability.

#### 4.4.13. *Delay scheduler*

Fair scheduling [20,21] policy was designed for fair sharing of capacity to all users, but there are two problems: sticky slot and head-of-line scheduling, which arises at the time of fair sharing. To overcome these two problems, the delay scheduler is developed. This scheduling [20] is introduced by applying changes to MapReduce with data locality to achieve better performance and lowest response time for the Map task. Facebook uses the same waiting approach to achieve locality in the Hadoop cluster. According to this policy, if data is not available for a task, then the task tracker will wait for a fixed amount of time. The scheduler checks the size of the job to see if there is any request for task allocation from a local node. The scheduler will skip too short jobs and look for any available subsequent job to run. If the process of job skipping continues for a long time, then it creates a non local task to avoid starvation. The delay scheduler solves the locality problems, which arise at the time of fair sharing such as sticky slot and head-of-line scheduling.

#### 4.4.14. *Advantages*

1. Simple. 2. This scheduling has no overhead for complex calculations. 3. Scheduler also resolves the locality problem.

#### 4.4.15. *Disadvantages*

1. Delay scheduling technique is not effective when a majority of the tasks is much more than an average job. 2. There are limited slots per node.

#### 4.4.16. *Deadline constraint scheduler*

In this scheduling [21,69] strategy, the user specified deadline constraints at the time of scheduling the jobs ensures that the jobs scheduled for execution meets the deadline [16,17]. It focuses on the issues of deadlines and increasing system utilization. It deals with the deadline requirement by the cost model of job execution, which considers parameters such as input size of data, data distribution, map and reduce run time etc. [69] Whenever any job is scheduled, it is checked by the scheduler whether it will be completed within the time specified by the deadline or not.

#### 4.4.17. *Advantages*

1. Deadline scheduler focuses more on the optimization of Hadoop implementation [69]. 2. This scheduling technique also increases system utilization.

#### 4.4.18. *Disadvantages*

1. There is a restriction that the nodes, should be uniform in nature, which incurs cost 2. There are some restrictions or issues of deadline, which are specified by the user for each job.

#### 4.4.19. *Resource aware scheduler*

This scheduler [18,69] is used to minimize resource utilization. In other schedulers such as Fair scheduler, FIFO, and Capacity scheduler, it is necessary that the administrator first assigns jobs to a queue and then ensure sharing of resources manually. In this scheduling [67], various resources like network utilization, memory utilization, CPU utilization, IO utilization, and disk utilization are shared more effectively.

Today, this technique has become a research challenge in cloud computing. In this scheme, the scheduling is accomplished by two nodes named Master Node and Work Node, which are also known as Job Tracker and Task Tracker, respectively. The Job Tracker maintains lists of tasks allocated to each Task Tracker, states of Task Trackers in the cluster, and the queue of the currently running jobs, while the Task Tracker is responsible for the execution of each task configured with the maximum number of available slots.

Two resource metrics, ”free slot filtering” and ”dynamic free slot advertisement”, are used in this scheduling. The scheduler computes the number of slots dynamically using dynamic free slot advertisement acquired from every node instead of the fixed computational slots configured with each Task Tracker node. Dynamic free slot advertisement uses these free slots dynamically as per demand. In free slot filtering, the maximum number of slots per node are fixed. Based on resource availability, free slots ordering is performed for advertising.

#### 4.4.20. *Advantages*

1. Resource aware scheduler improves the performance of job management 2. This scheduler also has better resource utilization in a cluster [69].

#### 4.4.21. *Disadvantages*

1. This scheduler does not provide support for the preemption of reduce tasks. 2. In this scheduler technique, there is a need for additional capabilities to manage network bottlenecks.

#### 4.4.22. *Matchmaking scheduler*

Matchmaking scheduling [84] focuses on the enhancement of data locality of map tasks. This scheduling gives equal chance to each slave node to seize local tasks before giving the slave node a non local task. A task is called local task when it is executed on the node where its data is available. This scheduler searches for matches, for example, in the case when slave node contains some input data, each Map task is not assigned. Each node is marked by a locality marker to confirm that every node gets equal opportunity to seize local tasks. This scheduling has the lowest response time but the highest data locality for map tasks. It has a moderately strict job scheduling order for job execution, unlike that in FIFO scheduling policy.

#### 4.4.23. *Advantages*

1. The matchmaking scheduling technique achieves the highest data locality. 2. This scheduler also provides high cluster utilization.

From the above comparison table, we conclude that Fair and Capacity schedulers are designed for short jobs as well as to resolve the fairness issue in scheduling, which involves providing a fair amount of resources to each job in the scheduling process. The Delay scheduling algorithm is designed to resolve locality issues, while Matchmaking scheduling algorithm is designed to resolve locality issues as well as for better cluster utilization. LATE scheduler optimizes the performance of jobs and minimizes job response time while processing the jobs. It has been proved to be highly robust with regard to heterogeneity. The Deadline constraint scheduler is designed for the optimization of Hadoop implementation. It focuses on the optimization of Hadoop implementation and ensures that the jobs scheduled for execution meets the deadline specified by the user. The Resource Aware scheduler is designed to obtain better resource utilization of clusters and better performance of jobs. A large amount of energy is required in the cluster data center when the Map and Reduce tasks are being processed. The Energy Aware scheduler is designed to optimize the energy utilization in the cluster data center. FIFO, Fair, Capacity, Delay, and Matchmaking schedulers work in homogeneous environments, while LATE, deadline Constraint, Resource Aware, and Energy Aware schedulers can work in both homogeneous as well as non homogeneous environments. Fig. 5 shows the details.

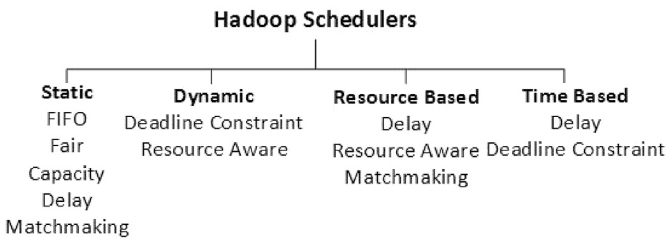

**Fig. 5.** Classification of Hadoop job schedulers.

## 5. Testing of real life schedule

### 5.1. *Experiment setup*

Hadoop is a distributed framework [95] designed for the deployment of low cost hardware. It is written in Java and used to run large data sets on commodity hardware. Hadoop can be installed on a variety of operating systems such as Windows, Linux, and Mac OS. Here, Hadoop is installed on Cent OS. For the performance evaluation of a real life scheduler, we implement a setup consisting of four machines. Our setup consists of a total of four slave nodes; among them one will work as the master node and all the four machines are on the same network connected through a switch, so that they can communicate with each other. The configuration of our setup is shown in Fig. 6.

Configuration of each machine:
Operating system: Cent OS 6.3
RAM: 4 GB
ROM: 500 GB
Processor: Intel i3

#### 5.1.1. *Scheduler configuration*

There are some changes that need to be configured for running schedulers in the Hadoop environment. FIFO is a default scheduler, and therefore, there is no need to configure this scheduler. We have to configure only the Fair and capacity schedulers.

#### 5.1.2. *Configuration of fair scheduler*

The following steps need to be performed to run Fair scheduler in the Hadoop environment.

1. Fair scheduler is available in the directory contrib/fairscheduler as a jar file, and the jar file in the directory appears as Hadoop-*-fair-scheduler.jar. Therefore, to use the fair scheduler, this jar file needs to be placed in the CLASSPATH.

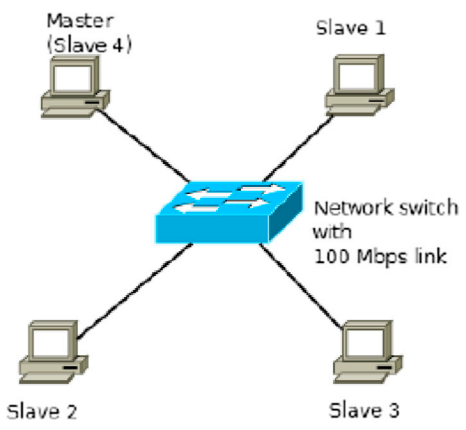

**Fig. 6.** Architecture of Hadoop Cluster testing Environment.

2. The property and its indicated value will be set as mapred, job-tracker.taskScheduler and org. apache.hadoop.mapred, FairScheduler, respectively.
3. Preemption should be on.
4. Weight booster should also be on to complete short jobs quickly.
5. The allocation file of fair scheduler should be specified to assign the resources to queues.

### 5.1.3. *Configuration of capacity schedulers*

The following steps need to be followed to run the Capacity scheduler in Hadoop environment.

1. The Capacity scheduler is available in the directory contrib/capacity-scheduler as a jar file and the jar file in the directory appears as hadoop-capacity-scheduler-*.jar. Therefore, to use the Capacity scheduler, this jar file needs to be placed in the CLASSPATH.
2. The property and its indicated value will be set as mapred. job-tracker.taskScheduler and org. apache.hadoop mapred. Capacity-TaskScheduler, respectively.

3. Create multiple queues with the capacity scheduler so that user can submit jobs in them. To set up the queues, we use mapred. queue.names property in config/Hadoop-site. xml.

4. Two queues have to be formed for job processing with capacities of 10–90% and 50-50%. Table 1 shows the details.

**Table 2**
Hadoop parameters.

| Parameters | Details (Configured Value) |
|---|---|
| HDFS block size | 64 MB |
| Speculative execution | Enabled |
| Heartbeat interval | 3 s |
| No of map tasks per node | 2 |
| No of reduce tasks per node | 1 |
| Replication factor | 2 |

### 5.1.4. *Experimental environment*

Here, our experiment environment is described so that one can easily understand our work. As discussed previously, our experiment setup consists of four machines, among which one works as a master node and all four works as a slave node. Each machine have 4 GB RAM, 500 GB ROM, Intel i3, 3.4 GHz processor, and 100 mbps network connection. The master node is responsible for running the Job Tracker and Name Node while slave node is responsible to run Task Tracker and Data Node. The nodes were installed with Cent OS 6.3 and Open JDK 1.6.0.24, and the execution environment was Hadoop 1.2.1. The configuration parameters of Hadoop are described in Table 2.

### 5.1.5. *Parameters for performance measure*

We use two representative applications in our experiment for performance evaluation: WordCount and Grep. WordCount has been

**Table 1**
Comparison of various Hadoop schedulers used in Big Data.

| Scheduler | Main Idea | Allocation of Job | Resources Sharing | Priority in Job Queue | Environment Homogeneous/ Heterogeneous | Features | Drawbacks |
|---|---|---|---|---|---|---|---|
| FIFO | First preference to job coming earlier, then to the job coming later. | Static | No | No | Homogeneous | Simple to implement and efficient among all schedulers. | Reduces data locality and Starvation of jobs, Long-running process will delay all jobs behind it. |
| Fair | Allocate equal shear of resources to each job. | Static | Yes | Yes | Homogeneous | Fair and dynamic resource reallocation, Fast response time to small jobs compared to large jobs. | Complicated configurations, does not consider weight of each job, which leads to unequal execution in each pool. |
| Capacity | creates many queues and each queue has a number of map and reduce slots, which are configurable. | Static | Yes | By default no | Homogeneous | Maximize the utilization of resources and throughput in cluster environment, Gives guarantee to reuse the unused capacity of jobs within queues. | Most complex among the the first three schedulers in this table, Difficulty in choosing queues properly. |
| LATE | find out process running with slow speed in the cluster and create equivalent process in the background as a backup | Static | Yes | Yes | Both | Optimizes the performance of jobs and minimizes job response time as much as possible, highly robust in the sense of heterogeneity. | Does not ensure reliability. |
| Delay | Achieve locality using waiting approach in the Hadoop cluster by relaxing fairness. | Static | No | Yes | Homogeneous | Simple, No overhead for complex calculations, Resolves the locality problem. | Not effective when majority of the tasks is bigger than an average job, few slots per node. |
| Matchmaking | gives equal chance to each slave node to seize local tasks before assigning slave node with non local tasks. | Static | Yes | Yes | Homogeneous | Provides highest rate of data locality, High utilization of clusters. | – |
| Deadline constraints | Constraints are specified by user while scheduling the jobs. | Dynamic | Yes | Yes | Both | Focus on Optimization of Hadoop implementation, increase system utilization | Restriction on Nodes, which incurs cost, should be uniform in nature. |
| Resource aware | Uses two resource metrics: Free slot filtering and Dynamic free slot advertisement. | Dynamic | Yes | Yes | Both | Better performance of job, better resource utilization in cluster. | Need for additional capabilities to maintain network bottlenecks. |
| Energy aware | Minimize energy consumption during execution of MapReduce job. | Dynamic | Yes | Yes | Both | Optimization of Energy. | Multiple Map Reduce jobs. |

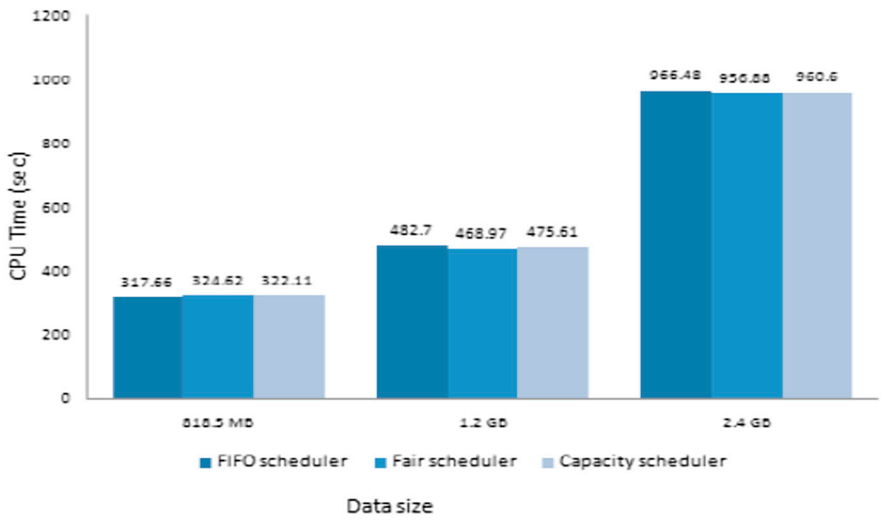

**Fig. 7.** Performance measure of WordCount job for different data sizes in terms of CPU Time for FIFO, Fair, and Capacity schedulers.

considered as a benchmark application for MapReduce. Many researcher have used it in their experiments. It reads text files and counts the number of words in those files. Grep is an application used in data mining. We use input text files of sizes 2.4 GB, 1.2 GB, and 818.5 MB. Different sizes of text will be used to find out the effect of data on the scheduler's performance. We use the following performance measures in our experiment.

CPU Time [48]: Time used by the CPU to execute the instructions of the program.

Turnaround Time [92]: Total time required from submission of job until the end of execution.

Data processing per second [89]: Input data size divided by application running time. The numerical values of all these parameters can be obtained from the job tracker web interface available with the Hadoop framework.

## 5.2. Experimental results

Here we analyse the performance of three schedulers, namely FIFO, Fair, and Capacity schedulers with WordCount and Grep applications for different data sizes.

### 5.2.1. Performance analysis based on CPU time

Fig. 7 shows the effect of data size on performance of job scheduling algorithms. For large data sizes (1.2 GB and 2.4 GB), it is clear from Fig. 7 that among FIFO, Fair, and Capacity schedulers, FIFO scheduler takes the highest CPU time to complete the jobs while Fair Scheduler takes the lowest time to process the same amount of data. For less data size (818.5 GB)FIFO takes less CPU Time than Fair and Capacity schedulers.

CPU time also depends on the application running on the system to process the data. In Fig. 8, we compare the WordCount and Grep applications in terms of the time required to process the 2.4 GB data. The result shows that WordCount takes higher a CPU time than Grep to process the same amount of data, i.e. 2.4 GB. In the case of WordCount application, FIFO takes more CPU time than Fair and Capacity schedulers, while in the case of Grep application, FIFO takes less CPU time than Fair and Capacity schedulers.

### 5.2.2. Performance analysis based on Turnaround time

Performance analysis is done based on data sizes with the Grep application. From the result shown in Fig. 9, Fair scheduler reduced the Turnaround time for all the different data sizes. For the large data size (2.4 GB), Fair and Capacity schedulers required less Turnaround time than FIFO scheduler.

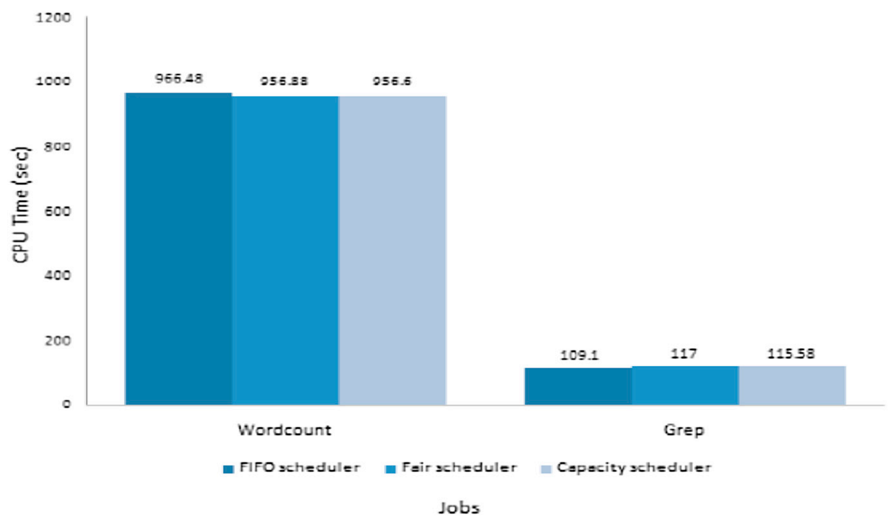

**Fig. 8.** Performance measure of WordCount and Grep applications in processing 2.4 GB data for FIFO, Fair, and Capacity schedulers.

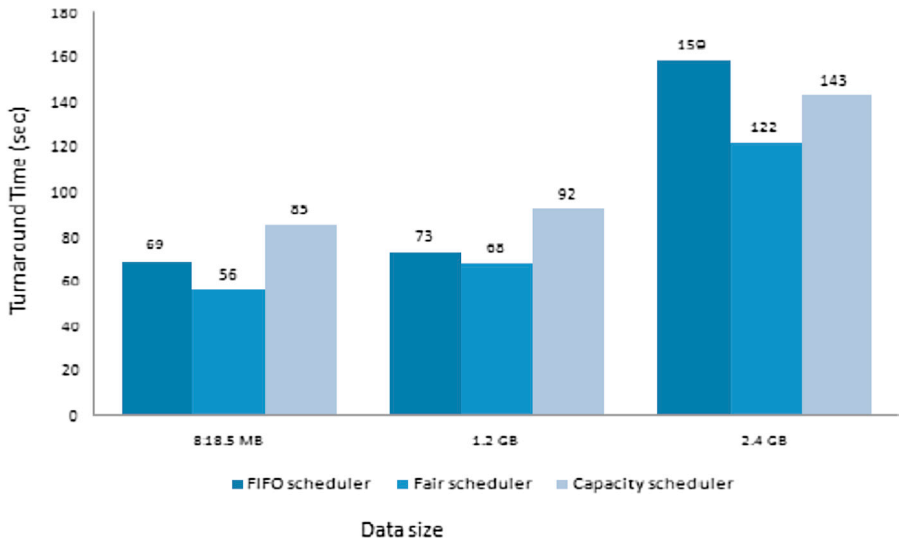

**Fig. 9.** Performance measure of Grep application for different data sizes in terms of Turnaround Time for FIFO, Fair, and Capacity schedulers.

### 5.2.3. *Performance analysis based on data processing per second*

The performance analysis based on data processing per second was conducted using different amounts of data. As shown in Fig. 10, for large amounts of data (1.2 GB and 2.4 GB), the FIFO scheduler processed less bytes then the Fair and Capacity schedulers, while in the case of less amount of data (818.5 MB), the Fair scheduler processed less bytes per second and the Capacity scheduler processed more bytes per second than the FIFO scheduler.

The type of application also matters in the processing of data. The data processing per second depends more or less on the processing applications used. As seen from the results in Fig. 11, WordCount processes very less data per second, compared to Grep. Grep processes approximately 8 times more data per second than WordCount. There is very little variation in the data processed per second for FIFO, Fair, and Capacity schedulers with WordCount, whereas with Grep this variation is a little bit more pronounced among FIFO, Fair, and Capacity schedulers.

## 6. Directions for future research

Although many literatures exist on job scheduling algorithms for the Hadoop environment, some of the significant requirements in this field still needs to be examined. Here, we provide some suggestions for future research, identified while studying those papers. According to the papers that were studied, we can divide future research into two major directions. The first direction concerns with the overall enhancement of job scheduling algorithms. The other direction concerns with the improvement of energy efficiency of Hadoop MapReduce.

### 6.1. *Direction for future research with regard to scheduling algorithms*

A critical open issue in scheduling algorithms is that different jobs have different quality requirements. Many jobs involve a balance with the other requirements. Only few algorithms have been designed to meet these multiple requirements. Thus, there is a lack of algorithms that consider the tradeoff among multiple quality requirements. We suggest that a thorough investigation needs to be conducted for the development of such algorithms in order to optimize multivariate problems by maximizing or minimizing one quality attribute while imposing a constraint on other quality attributes, so that it is easy to schedule different works on different resources.

An avenue for future research could also be to perform data placement on workload analysis for shared cluster and to design scheduling algorithms accordingly. Such combination of scheduling with data distribution strategies can help to improve the overall performance by

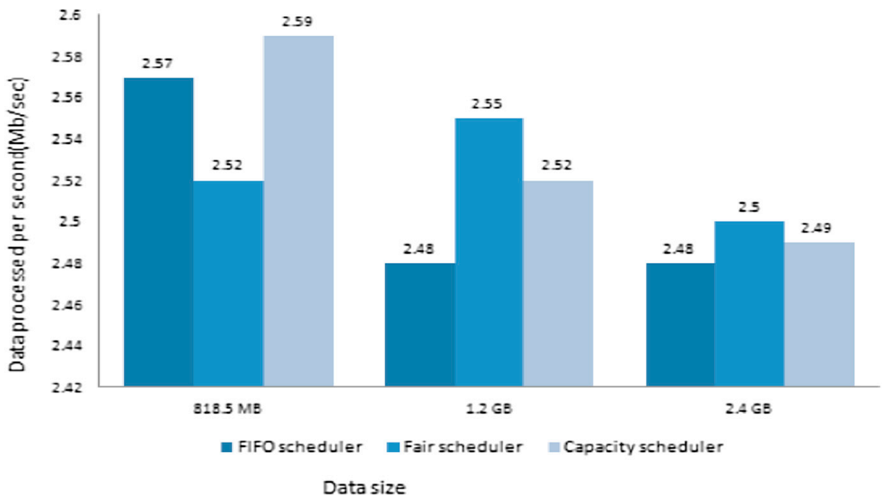

**Fig. 10.** Performance measure of WordCount application in terms of data processed per second for FIFO, Fair, and Capacity schedulers.

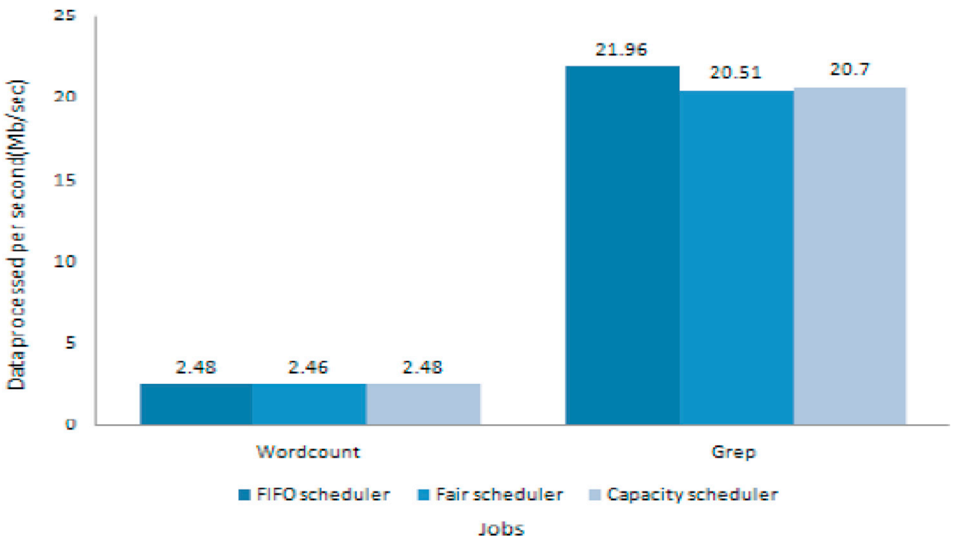

**Fig. 11.** Performance measure of WordCount and Grep applications in data processing of 2.4 GB for FIFO, Fair, and Capacity schedulers.

reducing the interference of segregated workloads. Hence, different workload analyses and data placement techniques need to be studied in detail.

### *6.2. Direction for future research to improve the energy efficiency improvement of Hadoop MapReduce*

Some research has also been done on energy aware schedulers on both low power and high performance machines, and it was observed that the CPU consumes half of the total power for the workload, and the power ranges of the two types of machines were very different. In future, we are considering the minimization of energy consumption by tuning the CPU frequency without affecting performance. The direction for parameters and their interaction will depend on the type of workload. Researchers can also perform experiments with varying workload to characterize the energy and performance in consideration of the known parameters. Predictive models can be made with these parameters, which improve energy efficiency by enabling fine tuning in systems.

The performance of a job in a cluster depends on parameters such as cluster configuration, job configuration, input data, and parallelism. Parallelism directly affects the performance of a job. Therefore, to the better performance, we need to increase parallelism by increasing the number of nodes in a cluster. For the performance analysis of the scheduler, more complex workload can be used. Additional configurable parameters such as the weights of pools and jobs can be used to improve the performance of fair scheduler. The number of simultaneous jobs per queue can be controlled to improve the performance of capacity scheduler. Therefore, these avenues can be used to improve the performance of jobs in autonomic computing, in which the job scheduling policy at the master node can be adjusted depending on the type of workload and changing cluster environment.

## 7. Discussion and conclusion

This paper summarized a comprehensive survey on big data and job scheduling algorithms in Hadoop environment. Job scheduling is a key factor to acquire high performance in big data processing. Various issues in big data include Data volume, Data variety, Data velocity, Security and Privacy, Cost, Connectivity and Data sharing, etc. For handling these issues, various job schedulers have been designed. This paper presents a comparative study on various job schedulers for big data processing in Hadoop environment such as FIFO, Delay, Fair, Capacity scheduling algorithm, etc. Each scheduler considers resources such as CPU, Memory, user constraints, IO, etc. The features and drawbacks of each scheduler are also discussed. From the comparative study and experiment results, we conclude that Fair and Capacity schedulers are designed for short jobs and equal utilization of resources, while Delay and matchmaking scheduling is designed to reduce locality issues. LATE, Deadline, and Resource Aware schedulers can be used in both homogeneous as well as heterogeneous environments. We also highlighted various parameters that affect the performance of job processing in Hadoop MapReduce such as Name Node, Data Node, Job Tracker, Task Tracker, Hadoop distributed file system, etc. Experiment results show that among FIFO, Fair, and Capacity schedulers, FIFO scheduler takes the highest CPU time to complete the jobs, while Fair Scheduler takes the lowest time to process the same amount of data. Fair scheduler reduces the turnaround time for varying data sizes, while for large data sizes, Fair and Capacity schedulers take less Turnaround time then FIFO scheduler. In the case of data processed per second, for large data sizes, FIFO scheduler processed less bytes per second than Fair and Capacity schedulers, while for less data sizes, Fair scheduler processed less bytes and Capacity scheduler processed more bytes per second than FIFO scheduler.

## Acknowledgement

This work is supported by the Applied Basic Research Program funded by Wuhan Science and Technology Bureau (Grant No. 2017010201010118).

## References

[1] B.T. Rao, L.S.S. Reddy, Survey on improved scheduling in hadoop MapReduce in cloud environments, Int. J. Comput. Appl. 34 (9) (2012) 29–33.
[2] S. Min Chen, Y. Liu Mao, Big data: a survey, ACM/Springer Mob. Netw. Appl. 19 (2) (April 2014) 171–209.
[3] K. Hwang, Min Chen, Big Data Analytics for Cloud/IoT and Cognitive Learning, Wiley, U.K, 2017.
[4] F. Xu, Y. Li, Min Chen, S. Chen, Mobile cellular big data: linking cyberspace and physical world with social ecology, IEEE Netw. 30 (3) (2016) 6–12.
[5] K. Hwang, Min Chen, J. Wu, Mobile big data management and innovative applications (editorial), IEEE Trans. Serv. Comput. 9 (5) (2016) 784–785.
[6] Y. Li, Min Chen, W. Dai, M. Qiu, Energy optimization with dynamic task scheduling mobile cloud computing, IEEE Syst. J. (2015), http://dx.doi.org/10.1109/JSYST.2015.2442994.
[7] Dazhao Cheng, Xiaobo Zhou, Palden Lama, Jun Wu, Changjun Jiang, Cross-platform resource scheduling for spark and MapReduce on YARN, IEEE Trans. Comput. PP (99) (2017), http://dx.doi.org/10.1109/TC.2017.2669964, 1–1.
[8] Albert Reuther, Chansup Byun, William Arcand, David Bestor, Bill Bergeron, Matthew Hubbell, Michael Jones, Peter Michaleas, Andrew Prout, Antonio Rosa,

Jeremy Kepner, Scheduler technologies in support of high performance data analysis, IEEE High. Perform. Extreme Comput. Conf. (HPEC) (2016) 1–6, http://dx.doi.org/10.1109/HPEC.2016.7761604.

[9] Sarah Shaikh, YARN versus MapReduce — a comparative study, in: Deepali Vora 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 2016, pp. 1294–1297.

[10] Ciprian Barbieru, Florin Pop, Soft real-time hadoop scheduler for big data processing in smart cities, in: IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), 2016, pp. 863–870, http://dx.doi.org/10.1109/AINA.2016.122.

[11] Uthira kumari, P. Asha, Hybrid scheduler to overcome the negative impact of job preemption for heterogeneous Hadoop systems, in: International Conference on Circuit, Power and Computing Technologies (ICCPCT), 2016, pp. 1–5, http://dx.doi.org/10.1109/ICCPCT.2016.7530206.

[12] Jyoti V. Gautam, Harshad kumar B. Prajapati, Vipul K. Dabhi, Sanjay Chaudhary, A survey on job scheduling algorithms in big data processing, IEE Conf. Pap. (March 2015), http://dx.doi.org/10.1109/ICECCT.2015.7226035.

[13] Mario Pastorelli, Damiano Carra, Matteo Dell Amico, Pietro Michiardi, HFSP: bringing size-based scheduling to hadoop, IEEE Trans. Cloud Comput. 5 (1) (2017) 43–56, http://dx.doi.org/10.1109/TCC.2015.2396056.

[14] Jianhong Zhai, Hongli Zhang, Xiaorou Zhong, Wei Li, Lai Wang, Zeyu He, Energy-efficient hadoop green scheduler, in: IEEE First International Conference on Data Science in Cyberspace (DSC), 2016, pp. 335–340, http://dx.doi.org/10.1109/DSC.2016.11.

[15] Yanling Shao, Chunlin Li, Wenyong Dong, Yunchang Liu, Energy-aware dynamic resource allocation on hadoop YARN cluster, in: IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)P-364–371, 2016, http://dx.doi.org/10.1109/HPCC-SmartCity-DSS.2016.0059.

[16] S. Rashmi, Basu Anirban, Deadline constrained Cost Effective Workflow scheduler for Hadoop clusters in cloud datacenter, in: International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), 2016, pp. 409–415, http://dx.doi.org/10.1109/CSITSS.2016.7779395.

[17] Norman Lim, Shikharesh Majumdar, Peter Ashwood-Smith, MRCP-RM: a technique for resource allocation and scheduling of MapReduce jobs with deadlines, IEEE Trans. Parallel Distributed Syst. PP (99) (2016), http://dx.doi.org/10.1109/TPDS.2016.2617324, 1–1.

[18] Zhihong Liu, Qi Zhang, Reaz Ahmed, Raouf Boutaba, Yaping Liu, Zhenghu Gong, Dynamic resource allocation for MapReduce with partitioning skew, IEEE Trans. Comput. 65 (11) (2016) 3304–3317, http://dx.doi.org/10.1109/TC.2016.2532860.

[19] Yang Liu, Yukun Zeng, Xuefeng Piao, High-responsive scheduling with MapReduce performance prediction on hadoop YARN, in: IEEE 22nd International Conference on Embedded and Real-time Computing Systems and Applications (RTCSA), 2016, pp. 238–247, http://dx.doi.org/10.1109/RTCSA.2016.51.

[20] Qiaomin Xie, Mayank Pundir, Yi Lu, Cristina L. Abad, Roy H. Campbell, Pandas: robust locality-aware scheduling with stochastic delay optimality, IEEE/ACM Trans. Netw. PP (99) (2016) 1–14, http://dx.doi.org/10.1109/TNET.2016.2606900.

[21] Xiangming Dai, Brahim Bensaou, Scheduling for response time in hadoop MapReduce, IEEE Int. Conf. Commun. (ICC) (2016) 1–6, http://dx.doi.org/10.1109/ICC.2016.7511252.

[22] Ran Zheng, Genmao Yu, Hai Jin, Xuanhua Shi, Qin Zhang, Conch: a cyclic MapReduce model for iterative applications, 24th Euromicro Int. Conf. Parallel Distributed, Network-Based Process. (PDP) (2016) 264–271, http://dx.doi.org/10.1109/PDP.2016.66.

[23] Thomas C. Bressoud, Qiuyi Tang, “Results of a model for hadoop YARN MapReduce tasks, IEEE Int. Conf. Clust. Comput. (2016) 443–446, http://dx.doi.org/10.1109/CLUSTER.2016.51 (CLUSTER).

[24] A.A. Safaei, Real-time processing of streaming big data, Real-Time Syst. 53 (1) (2017), http://dx.doi.org/10.1007/s11241-016-9257-0.

[25] M. Cavallo, L. Cusmà, G. Di Modica, C. Polito, O. Tomarchio, A scheduling strategy to run hadoop jobs on geodistributed data, in: A. Celesti, P. Leitner (Eds.), Advances in Service-oriented and Cloud Computing. ESOCC Workshops 2015. Communications in Computer and Information Science vol. 567, Springer, Cham, 2016.

[26] X. Cai, F. Li, P. Li, et al., SLA-aware energy-efficient scheduling scheme for Hadoop YARN, J. Supercomput. (2016), http://dx.doi.org/10.1007/s11227-016-1653-7.

[27] Deepak Vohra, Practical Hadoop Ecosystem, Apress, springer, 30 September 2016, pp. 3–162, http://dx.doi.org/10.1007/978-1-4842-2199-0-1. Print ISBN 978-1-4842-2198-3, Online ISBN 978-1-4842-2199-0.

[28] J. Li, S. Shi, H. Wang, Optimization analysis of hadoop, in: W. Che, et al (Eds.), Social Computing. ICYCSEE 2016. Communications in Computer and Information Science vol. 623, Springer, Singapore, 2016.

[29] K. Bok, J. Hwang, J. Lim, et al., An efficient MapReduce scheduling scheme for processing large multimedia data, Multimedia Tools Appl. (2016), http://dx.doi.org/10.1007/s11042-016-4026-6.

[30] Bunjamin Memishi, Shadi Ibrahim, María S. Pérez, Gabriel Antoniu, “Fault Tolerance in MapReduce: a Survey,” Resource Management for Big Data Platforms, Part of the Series Computer Communications and Networks, Springer International Publishing, pp 205–240, DOI 10.1007/978-3-319-44881-7-11, Print ISBN 978-3-319-44880-0, Online ISBN 978-3-319-44881-7.

[31] I.A.T. Hashem, N.B. Anuar, A. Gani, et al., MapReduce: review and open challenges, Scientometrics 109 (2016) 389, http://dx.doi.org/10.1007/s11192-016-1945-y.

[32] L. Rodríguez-Mazahua, C.A. Rodríguez-Enríquez, J.L. Sánchez-Cervantes, et al., A general perspective of Big Data: applications, tools, challenges and trends, J. Super Comput. 72 (2016) 3073, http://dx.doi.org/10.1007/s11227-015-1501-1.

[33] Y. Liu, M. Qiu, C. Liu, et al., Big data challenges in ocean observation: a survey, Personal Ubiquitous Comput. 21 (2017) 55, http://dx.doi.org/10.1007/s00779-016-0980-2.

[34] I. Anagnostopoulos, S. Zeadally, E. Exposito, Handling big data: research challenges and future directions, J. Super Comput. 72 (2016) 1494, http://dx.doi.org/10.1007/s11227-016-1677-z.

[35] M. Brahmwar, M. Kumar, G. Sikka, Tolhit – a scheduling algorithm for hadoop cluster, Orig. Res. Article Procedia Comput. Sci. 89 (2016) 203–208.

[36] O. Yildiz, S. Ibrahim, G. Antoniu, Enabling fast failure recovery in shared Hadoop clusters: towards failure-aware scheduling, Future Gener. Comput. Syst. 74 (2017) 208–219.

[37] Wenhong Tian, Zhao Yong, 9-Energy efficiency scheduling in hadoop, Optim. Cloud Resour. Manag. Sched. (2015) 179–204.

[38] Guilherme W. Cassales, Andrea S. Charão, Manuele Kirsch Pinheiro, Carine Souveyet, Luiz A. Steffenel, Context-aware scheduling for apache hadoop over pervasive environments, Procedia Comput. Sci. 52 (2015) 202–209.

[39] S. Suresh, N.P. Gopalan, An optimal task selection scheme for hadoop scheduling, IERI Procedia 10 (2014) 70–75.

[40] Ilias Mavridis, Helen Karatza, Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark, J. Syst. Softw. 125 (March 2017) 133–151.

[41] Miguel Liroz-Gistau, Reza Akbarinia, Divyakant Agrawal, Patrick Valduriez, FP-Hadoop: efficient processing of skewed MapReduce jobs, Orig. Res. Article, Inf. Syst. 60 (August–September 2016) 69–84.

[42] Xiao Ling, Yi Yuan, Dan Wang, Jiangchuan Liu, Jiahai Yang, Joint scheduling of MapReduce jobs with servers: performance bounds and experiments, J. Parallel Distributed Comput. 90–91 (April 2016) 52–66.

[43] Xiaoping Li, Tianze Jiang, Rubén Ruiz, Heuristics for periodical batch job scheduling in a MapReduce computing framework, Inf. Sci. 326 (1) (January 2016) 119–133.

[44] Jiong Xie, FanJun Meng, HaiLong Wang, HongFang Pan, JinHong Cheng, Xiao Qin, Research on scheduling scheme for hadoop clusters, Procedia Comput. Sci. 18 (2013) 2468–2471.

[45] Yuliang Shi, Kaihui Zhang, Lizhen Cui, Lei Liu, Yongqing Zheng, Shidong Zhang, Han Yu, MapReduce short jobs optimization based on resource reuse, Microprocess. Microsyst. vol. 47 (November 2016) 178–187. Part a.

[46] Victoria J. Hodge, Simon O'Keefe, Jim Austin, Hadoop neural network for parallel and distributed feature selection, Neural Netw. 78 (June 2016) 24–35.

[47] J. Geetha, N. UdayBhaskar, P. ChennaReddy, Data-local reduce task scheduling, Procedia Comput. Sci. 85 (2016) 598–605.

[48] Shadi Ibrahim, Tien-Dat Phan, Alexandra Carpen-Amarie, Houssem-Eddine Chihoub, Diana Moise, Gabriel Antoniu, Governing energy consumption in Hadoop through CPU frequency scaling: an analysis, Future Gener. Comput. Syst. 54 (January 2016) 219–232.

[49] Uthayasankar Sivarajah, Muhammad Mustafa Kamal, Zahir Irani, Vishanth Weerakkody, Critical analysis of Big Data challenges and analytical methods, J. Bus. Res. 70 (January 2017) 263–286.

[50] Lina Zhou, Shimei Pan, Jianwu Wang, Athanasios V. Vasilakos, Machine learning on big data: opportunities and challenges, Neurocomputing 237 (10 May 2017) 350–361.

[51] Lisia S. Dias, Marianthi.G. Ierapetritou, Integration of scheduling and control under uncertainties: review and challenges, Chem. Eng. Res. Des. 116 (December 2016) 98–113.

[52] Natalija Koseleva, Guoda Ropaite, Big data in building energy efficiency: understanding of big data and main challenges, Procedia Eng. 172 (2017) 544–549.

[53] Gema Bello-Orgaz, Jason J. Jung, David Camacho, Social big data: recent achievements and new challenges, Inf. Fusion 28 (March 2016) 45–59.

[54] S. Suresh, N.P. Gopalan, An optimal task selection scheme for hadoop scheduling, IERI Procedia 10 (2014) 70–75.

[55] Philip Derbeko, Shlomi Dolev, Ehud Gudes, Shantanu Sharma, Security and privacy aspects in MapReduce on clouds: a survey, Comput. Sci. Rev. 20 (May 2016) 1–28.

[56] Seema Maitrey, C.K. Jha, MapReduce: simplified data analysis of big data, Procedia Comput. Sci. 57 (2015) 563–571.

[57] David Gil, Il-Yeol Song, Modeling and management of big data: challenges and opportunities, Future Gener. Comput. Syst. 63 (October 2016) 96–99.

[58] Sachin Bende, Rajashree Shedge, Dealing with small files problem in hadoop distributed file system, Procedia Comput. Sci. 79 (2016) 1001–1012.

[59] Jie Zhang, Xiao Yao, Guangjie Han, Yiqi Gui, A survey of recent technologies and challenges in big data utilizations, in: International Conference on Information and Communication Technology Convergence (ICTC), 2015, pp. 497–499, http://dx.doi.org/10.1109/ICTC.2015.7354594.

[60] Huanle Xu, Wing Cheong Lau, Optimization for speculative execution in big data processing clusters, IEEE Trans. Parallel Distributed Syst. 28 (2) (2017) 530–545, http://dx.doi.org/10.1109/TPDS.2016.2564962.

[61] Min Chen, Shiwen Mao, Yin Zhang, Victor C.M. Leung, Big Data: Related Technologies, Challenges Future Prospects, Springer Cham Heidelberg New York Dordrecht London, 2014, ISBN 978-3-319-06244-0, pp. 16–18.

[62] S.J. Yang, Y.-R. Chen, Design adaptive task allocation scheduler to improve MapReduce performance in heterogeneous clouds, J. Netw. Comput. Appl. 57 (2015) 61–70, http://dx.doi.org/10.1016/j.jnca.2015.07.012.

[63] D. Yoo, K.M. Sim, A comparative review of job scheduling for MapReduce, in: Cloud Computing and Intel. Syst. (CCIS), IEEE Int. Conf. on. IEEE, 2011.

[64] A. Gani, A. Siddiqa, S. Shamshirband, F. Hanum, A survey on indexing techniques for big data: taxonomy and performance evaluation, Knowl. Inf. Syst. 46 (2) (2016) 241–284.
[65] M.D. Assuncao, R.N. Calheiros, S. Bianchi, M.A.S. Netto, R. Buyya, Big data computing and clouds: trends and future directions, J. Parallel Distributed Comput. 79–80 (2015) 3–15.
[66] S. Divya, R. Kanya Rajesh, Rini Mary Nithila I, Vinothini M, Big data analysis and its scheduling policy- hadoop, IOSR J. Comput. Eng. (IOSR-JCE) 17 (1) (2015) 36–40 e-ISSN: 2278-0661, p-ISSN: 2278-8727, Ver. IV. .
[67] Shyam Deshmukh, J.V. Aghav, Rohan Chakravarthy, Job classification for MapReduce scheduler in heterogeneous environment, in: International Conference on Cloud, Ubiquitous Computing and Emerging Technologies, 2013.
[68] Bo Wang, Jinlei Jiang, Yongwei Wu, Guangwen Yang, Keqin Li, Accelerating MapReduce on commodity clusters: an SSD-empowered approach, in: IEEE Transactions on Big Data, IEEE, 2016.
[69] Dazhao Cheng, Jia Rao, Changjun Jiang, Xiaobo Zhou, Resource and deadline-aware job scheduling in dynamic hadoop clusters, in: IEEE 29th International Parallel and Distributed Processing Symposium, 2015.
[70] Ibrahim Abaker, Targio Hashem, Ibrar Yaqoob, NorBadrul Anuar, Salimah Mokhtar, Abdullah Gani, Samee UllahKhan, The rise of the big data on cloud computing: review and open research issues, Inf. Syst. 47 (2015) 98–115 (IEEE).
[71] Bhavin J. Mathiya, Vinodkumar L. Desai, Apache hadoop Yarn parameter configuration challenges and optimization, in: International Conference on Soft-computing and Network Security (ICSNS -2015), Feb. 25-27, 2015.
[72] C. Sreedhar, N. Kasiviswanath, P. Chenna Reddy, A survey on big data management and job scheduling, Int. J. Comput. Appl. 130 (13) (November 2015) 0975–8887.
[73] Cameron Seay, Rajeev Agrawal, Anirudh Kadadi, Yannick Barel, Using hadoop on the mainframe:a big solution for the challenges of big data, in: 12th International Conference on Information Technology - New Generations, 2015.
[74] S. Suthaharan, Big data classification: problems and challenges in network intrusion prediction with machine learning, ACM SIGMETRICS Perform. Eval. Rev. 41 (4) (2014) 70–73.
[75] F. Afrati, S. Dolev, E. Korach, S. Sharma, J.D. Ullman, Assignment Problems of Different Sized Inputs in Map Reduce, 2015 arXiv:1507.04461.
[76] J.C. Anjos, I. Carrera, W. Kolberg, A.L. Tibola, L.B. Arantes, C.R. Geyer, MRA++: scheduling and data placement on MapReduce for heterogeneous environments, Future Gener. Comput. Syst. 42 (2015) 22–V35.
[77] B.T. Rao, L.S.S. Reddy, Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments, 2012 arXiv preprintarXiv:1207.0780.
[78] Hadoop Wiki, [online]. Available: http://wiki.apache.org/hadoop/GangliaMetrics.
[79] Apache Hadoop, [online]. Available: http://hadoop.apache.org.
[80] Hadoop Distributed File System,[online]. http://hadoop.apache.org/hdfs.
[81] Hadoop Tutorial, http://developer.yahoo.com/hadoop/tutorial/module1.html.
[82] Big Data Now: 2012 Edition, Cover Designer: Karen Montgomery, Interior Designer: David Futato, O'Reilly Media, Inc., 2009. ISBN: 978-1-449-35671-2.
[83] Y. Bu, B. Howe, M. Balazinska, M.D. Ernst, Hadoop: efficient iterative data processing on large clusters, Proc. VLDB Endow. 3 (1–2) (2010) 285–296.
[84] J.V. Gautam, H.B. Prajapati, V.K. Dabhi, S. Chaudhary, A survey on job scheduling algorithms in big data processing, in: IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT'15), Coimbatore, 2015, pp. 1–11.
[85] Z. Jia, R. Zhou, C. Zhu, L. Wang, W. Gao, Y. Shi, J. Zhan, L. Zhang, The. Implications of Diverse Applications and Scalable Data Sets in Benchmarking Big Data Systems. – in: Specifying Big Data Benchmarks, Springer, 2014, pp. 44–59.
[86] X. Shi, M. Chen, L. He, X. Xie, L. Lu, H. Jin, Y. Chen, S. Wu, Mammoth: gearing hadoop towards memory-intensive mapreduce applications," parallel and distributed systems, IEEE Trans. 26 (8) (2015) 2300–2315.
[87] S. Liu, J. Xu, Z. Liu, X. Liu, Evaluating task scheduling in hadoop-based cloud systems, in: 2013 IEEE International Conference on Big Data, IEEE, 2013, pp. 47–53.
[88] Bo Wang, Jinlei Jiang, Yongwei Wu, Guangwen Yang, Keqin Li, Accelerating MapReduce on commodity clusters: an SSD-empowered approach, IEEE Trans. Big Data (2016) 2332–7790.
[89] Ibrahim Abaker, Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, Samee Ullah Khan, The rise of big data, on cloud computing:Review and open research issues, Inf. Syst. 47 Elsevier (2015) 98–115.
[90] D. Park, B. Debnath, D. Du, A dynamic switching flash translation layer based on a page-level mapping, IEICE Trans. Inf. Syst. E99-D (6) (June 2016) 51–60.
[91] Dongchul Park, Yang-Suk Kee, In-storage computing for hadoop MapReduce framework: challenges and possibilities, IEEE Trans. Comput. (July 2015) 1.
[92] Zhigang Wang, Yanming Shen, Job-aware scheduling for big data processing, in: International Conference on Cloud Computing and Big Data, IEEE, 2015, 978-1-4673-8350-9/15.
[93] J.C. Lin, I.C. Yu, E.B. Johnsen, M.C. Lee, ABS-YARN: a formal framework for modeling hadoop YARN clusters, in: P. Stevens, A. wasowski (Eds.), Fundamental Approaches to Software Engineering. FASE 2016. Lecture Notes in Computer Science vol. 9633, Springer, Berlin, Heidelberg, 2016.
[94] C. Dong, Q. Shen, L. Cheng, Y. Yang, Z. Wu, SECapacity: a secure capacity scheduler in YARN, in: K.Y. Lam, C.H. Chi, S. Qing (Eds.), Information and Communications Security. ICICS 2016. Lecture Notes in Computer Science vol. 9977, Springer, Cham, 2016.
[95] P. Pant, R. Tanwar, An overview of big data opportunity and challenges, in: A. Unal, M. Nayak, D. Mishra, D. Singh, A. Joshi (Eds.), Smart Trends in Information Technology and Computer Communications. SmartCom 2016. Communications in Computer and Information Science vol. 628, Springer, Singapore, 2016.
[96] S. Cheng, B. Liu, Y. Shi, Y. Jin, B. Li, Evolutionary computation and big data: key challenges and future directions, in: Y. Tan, Y. Shi (Eds.), Data Mining and Big Data. DMBD 2016. Lecture Notes in Computer Science vol. 9714, Springer, Cham, 2016.
[97] META Group research note [online]. https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf.