REVIEW

MapReduce scheduling algorithms in Hadoop: a systematic study

Soudabeh Hedayati[1*], Neda Maleki[2], Tobias Olsson[2], Fredrik Ahlgren[2], Mahdi Seyednezhad[3] and Kamal Berahmand[4]

Abstract

Hadoop is a framework for storing and processing huge volumes of data on clusters. It uses Hadoop Distributed File System (HDFS) for storing data and uses MapReduce to process that data. MapReduce is a parallel computing framework for processing large amounts of data on clusters. Scheduling is one of the most critical aspects of MapReduce. Scheduling in MapReduce is critical because it can have a significant impact on the performance and efficiency of the overall system. The goal of scheduling is to improve performance, minimize response times, and utilize resources efficiently. A systematic study of the existing scheduling algorithms is provided in this paper. Also, we provide a new classification of such schedulers and a review of each category. In addition, scheduling algorithms have been examined in terms of their main ideas, main objectives, advantages, and disadvantages.

Keywords Distributed systems, Resource allocation, Scheduling algorithms, Hadoop, MapReduce, Fair scheduling

Introduction

Big data is a term used to describe a collection of extremely large amounts of data that cannot be handled by conventional database management tools [1]. Big data has become very popular in information technology, where data is becoming increasingly complicated and large amounts of it are being created every day. Data comes from social networking sites, business transactions, sensors, mobile devices, etc. [2]. Due to the increase in volume, velocity, and variety of data, processing leads to complexities and challenges. Thus, big data becomes a complex process in terms of correctness, transformation, matching, relating [3]. A new platform

*Correspondence:
Soudabeh Hedayati
hedayati.soodabeh@gmail.com
[1] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
[2] Applied IoT Lab, Department of Computer Science and Media Technology, Linnaeus University, Kalmar, Sweden
[3] School of Computing, Florida Institute of Technology, Melbourne, USA
[4] School of Computer Sciences, Science and Engineering Faculty, Queensland University of Technology (QUT), Brisbane, Australia

is required for data transmission, storage, and processing because the data is so big and unstructured. The platform can process and analyze huge volumes of data at a reasonable speed. This necessity led to the development of parallel and distributed processing in clusters and grades [4]. In order to hide the complexity of the parallel processing system from the users, numerous frameworks have been released [5].

MapReduce is a programming pattern that is popular among all frameworks. Using the Map and Reduce functions of MapReduce, users do not have to worry about the details of parallelism when defining parallel processes, such as data distribution, load balancing, and fault tolerance [6]. MapReduce is used to process high volumes of data concurrently. Map and Reduce are the two functions of MapReduce. In this framework, the first step in parallel computing is to assign map tasks to various nodes and perform them on input data. Then, the final results are generated by combining the map outputs and applying the reduce function [7, 8].

MapReduce is in competition with other program paradigms like Spark and DataMPI. The choice of MapReduce for investigation is based on the fact that

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 2 of 30

it is high-performance open source, utilized by many large companies to process batch jobs [9, 10], and is our future research area. Scheduling is the process of assigning tasks to the nodes, which is a critical factor for improving system performance in MapReduce. There are many algorithms to solve scheduling issues with different techniques and approaches. The main goal of scheduling is to increase throughput while reducing response time and improving performance and resource utilization [11, 12].

Using recent articles and inclusion and exclusion criteria, we provide a systematic and comprehensive survey of MapReduce scheduling papers. MapReduce scheduling was the subject of a systematic literature review in 2017 [13], but there hasn't been another systematic study. As far as we know, our review is the first systematic study from 2017 to June 2023. In this paper, we have reviewed the existing scheduling algorithms and identified trends and open challenges in this area. To answer the research questions (RQs), we gathered and analyzed relevant data from papers on MapReduce scheduling and provided the answers.

In this study, our purpose is to provide the results of a systematic survey on MapReduce scheduling algorithms. Thus, we consider current algorithms, compare the differences between the schedulers, and describe several scheduling algorithms. The remainder of the paper is structured as follows. "Background and research method" section has two parts: In part one, an architectural overview of MapReduce and Hadoop is introduced, and in part two, our research method is provided. "Regular surveys" section discusses schedulers in Hadoop MapReduce and categorizes them. It also presents a comparison of selected algorithms. In "Schedulers in Hadoop MapReduce" section, we discuss the mentioned schedulers and analyze the results. Finally, our conclusions are provided in "Discussion" section.

Background and research method

Background

Doug Cutting and Mike Cafarela designed Hadoop specifically for distributed applications. It is an open-source Apache project that used Google's MapReduce approach. Hadoop has two major parts: Hadoop Distributed File System (HDFS) and MapReduce [14], which is shown in Fig. 1.

HDFS: the data storage part of Hadoop is HDFS. The HDFS architecture is based on master/slave. It consists of one NameNode and several DataNodes. The NameNode acts as a master node, while the DataNodes act as a slave to the master node [16]. Also, NameNode maps input data splits to DataNodes and maintains the metadata, and DataNodes store application data. The data stored in HDFS is fetched by MapReduce for computation [15, 17].

MapReduce is the processing unit of Hadoop with two primary parts: one JobTracker and numerous TaskTrackers. The JobTracker breaks the job into several map and reduce tasks and assigns them to TaskTrackers, and the TaskTrackers run the tasks [18]. The MapReduce job execution flow is depicted in Fig. 2. Map tasks are performed in parallel on input splits to create a set of intermediate key-value pairs. Using key-value pairs, these pairs are shuffled across multiple reduce tasks. One key is accepted by each Reduce task at a time, and that key's data is processed, generating the output files which are stored on HDFS [19].

Research method

According to [20–22], the papers are selected using the following method:

- Several research questions are established based on the research area;
- Keywords are discovered based on the research questions;
- Search strings are defined according to the keywords;
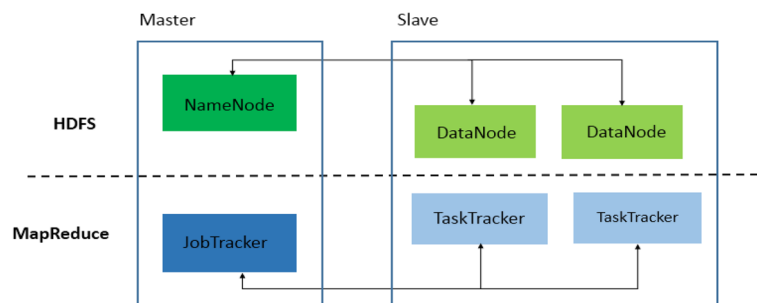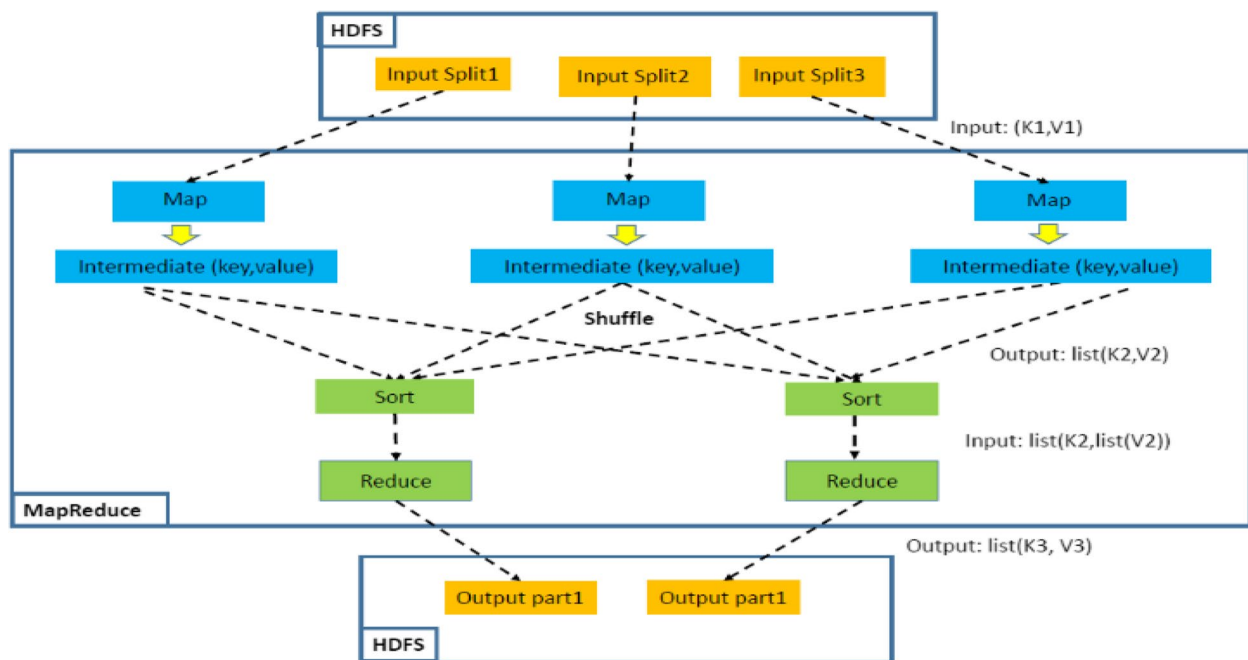- The final papers are vetted according to several inclusion and exclusion criteria.

Fig. 1 Hadoop Architecture [15]

Fig. 2 MapReduce job execution flow [19]

Research question
In this paper, we review and analyze the selected studies to present a summary of current research on MapReduce scheduling. We plan to respond to the following research questions:

- RQ1: What was the trend of publications in the field of Hadoop MapReduce scheduling in the last five years?
- RQ2: What are the proposed schedulers, and what are the techniques used?
- RQ3: What are the key ideas of each paper?
- RQ4: What evaluation techniques have been used to assess the results of each paper?
- RQ5: What scheduling metrics are considered?

Search strategy
We used search strings to find systematic mapping and literature studies. The search strings were defined after the study questions were created. The databases were also described. Research keywords: Using the research questions, we chose the keywords. The identified keywords were: "Hadoop", "MapReduce", and "Scheduling". Search strings: we defined the search strings using the selected keywords according to the research questions. The survey's initial results were found using the search string, i.e., "Hadoop OR MapReduce" AND

"Scheduling." This expressive style was subsequently adapted to suit the specific requirements of each database. Sources: After defining the search string, we selected the following databases as sources:

- IEEE Xplore
- Science Direct
- Springer
- ACM Portal

The search period was from 2009 to 2023, and it was carried out in July 2023. A typical attendance graph is shown in Fig. 3, with its characteristic funnel shape.
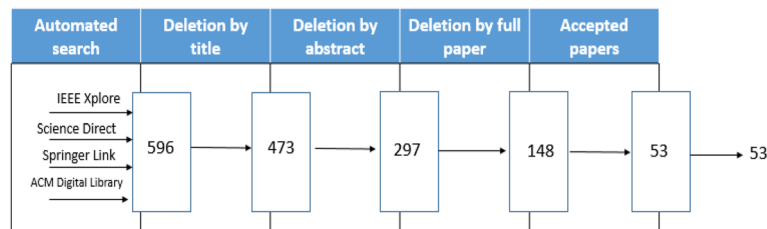
Search selection
After getting the database results, each paper must be carefully examined to ensure it pertains to our survey context. To find the studies, the following inclusion and exclusion criteria were applied. The results removed from each stage can be seen in Fig. 4.

Stage 1. Apply the search query to all the sources, gathering the results.
Stage 2. Apply inclusion/exclusion criteria to the paper title.
Stage 3. Apply inclusion/exclusion criteria to the paper abstract.
Stage 4. Apply inclusion/exclusion criteria to the introduction.

Fig. 3 References Word Cloud—This figure shows the word cloud generated by processing the references. Top 200 most used words are selected. The larger the words are, the more frequently they are used

Fig. 4 Selection of the papers

Inclusion criteria The relevance of a work to the research issues is one of the inclusion criteria. First, we examined the study titles and abstracts from the initial database searches. The introduction then conducted a second analysis of the chosen studies from this stage. The study was discarded whenever one of the inclusion criteria was not met. The inclusion criteria are as follows:

1) Studies are addressing scheduling concepts for Hadoop MapReduce.
2) Studies use predetermined performance measures to assess their performance approach.

3) Studies are presented at conferences or published in journals (peer review).

Exclusion criteria Studies were disqualified based on an analysis of their title, abstract, and, if necessary, introduction. The exclusion criteria are as follows:

1) Studies that do not address scheduling in Hadoop MapReduce.
2) Studies whose entire texts are not accessible at the source.
3) Studies unrelated to the research questions.

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 5 of 30

4) Repeated studies that were printed in multiple sources;
5) Presentations, demonstrations, or tutorials.

Data extraction

During the phase of data extraction for analysis of the selected studies, the data gathered from them are summarized. We synthesized the data to answer the research question RQ1 in this section.

Answer to Question RQ1 What was the trend of publications in the field of Hadoop MapReduce scheduling in the last five years?

Figure 5 shows the percentage of selected studies per year. As can be seen, 17 (34%) studies have been published in the last five years. To be more specific, 1 (2%) articles were published in 2023, 7 (14%) articles were published in 2022, 4 (8%) studies were published in 2021, 3 (6%) studies were published in 2020, and 2 (4%) articles were published in 2019.

Figure 6 shows the percentage of selected studies from each database. It is shown that 7% of studies are related to ACM Digital Library, 65% are related to IEEE Xplore, 5% belong to Science Direct, and 23% belong to Springer Link.

Figure 7 presents the percentage of studies in each type. As can be seen, 61% of the studies were presented at a conference, and 39% were published in journals.

Regular surveys

In this section, we aim to provide an extensive review of regular surveys conducted in the field. Our goal is to comprehensively analyze and summarize the existing surveys to offer a comprehensive understanding of the subject matter.

Ghazali et al. [23] presented a novel classification system for job schedulers, categorizing them into three distinct groups: job schedulers for mitigating stragglers, job schedulers for enhancing data locality, and job schedulers for optimizing resource utilization. For each job scheduler within these groups, they provided a detailed explanation of their performance-enhancing approach and conducted evaluations to identify their strengths and weaknesses. Additionally, the impact of each scheduler was assessed, and recommendations were offered for selecting the best option in each category. Finally, the authors provided valuable guidelines for choosing the most suitable job scheduler based on specific environmental factors. However, the survey is not systematic and the process of selecting articles is not clear. Moreover, there is no information about the environment and the platform for implementation or simulation of the surveyed articles.

Abdallat et al. [24] focused on the topic of job scheduling algorithms in Big Data Hadoop environment. The authors emphasize the importance of efficient job scheduling in processing large amounts of data in real-time, considering the limitations of traditional
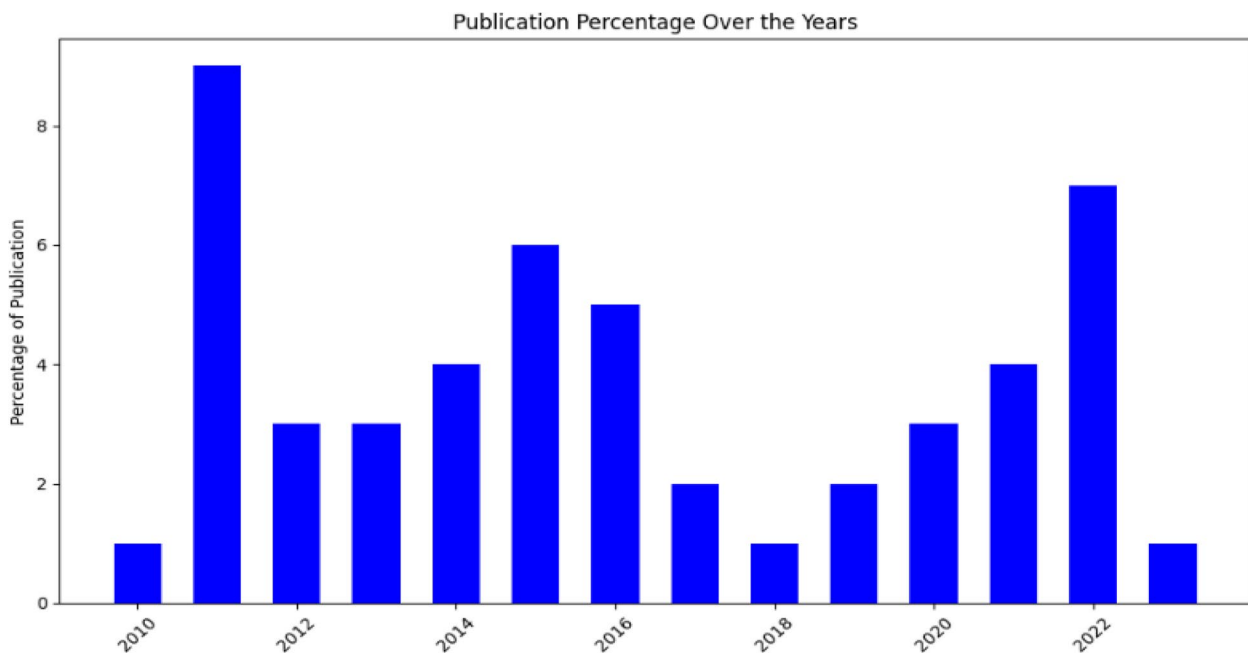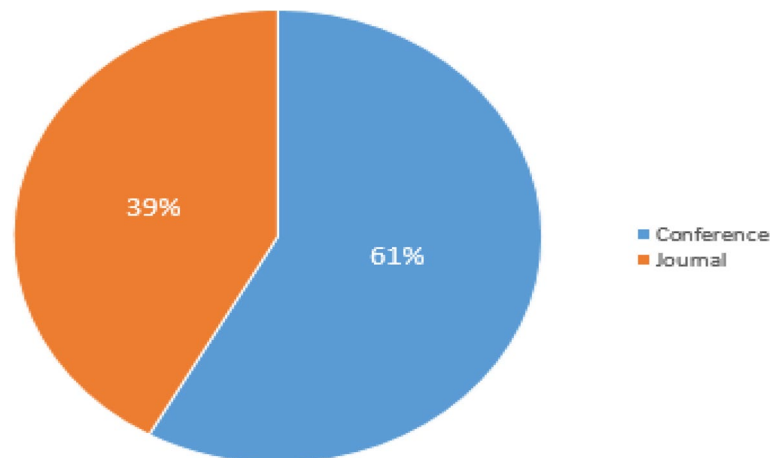

Fig. 5 Percentage of selected studies per year
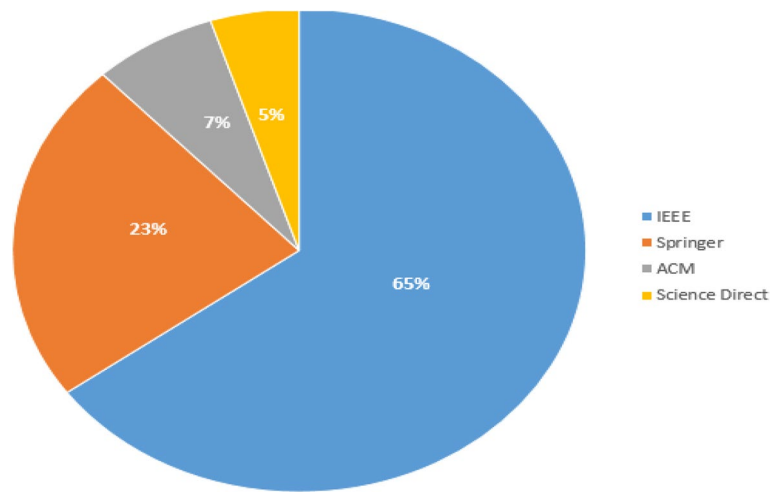
Fig. 6 Percentage of selected studies per database

Fig. 7 Percentage of selected studies per type database

ecosystems. The paper provides background information on the Hadoop MapReduce framework and conducts a comparative analysis of different job scheduling algorithms based on various criteria such as cluster environment, job allocation strategy, optimization strategy, and quality metrics. The authors present use cases to illustrate the characteristics of selected algorithms and offer a comparative display of their details. The paper discusses popular scheduling considerations, including locality, synchronization, and fairness, and evaluates Hadoop schedulers based on metrics such as locality, response time, and fairness. However, the survey is not systematic and the process of selecting articles is not clear. Also, there is no comparison between these studies' advantages and disadvantages, and there is no information about the environment and

the platform for implementation or simulation of the surveyed articles.

Hashem et al. [25] conducted a comprehensive comparison of resource scheduling mechanisms in three widely-used frameworks: Hadoop, Mesos, and Corona. The scheduling algorithms within MapReduce were systematically categorized based on strategies, resources, workload, optimization approaches, requirements, and speculative execution. The analysis encompassed two aspects: taxonomy and performance evaluation, where they thoroughly reviewed the advancements made in MapReduce scheduling algorithms. Additionally, the authors highlighted existing limitations and identified potential areas for future research in this domain. However, the survey is not systematic and the process of selecting articles is not clear. Also, there is

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 7 of 30

no information about the environment and the platform for implementation or simulation of the surveyed articles.

Soualhia et al. [26] analyzed 586 papers to identify the most significant factors impacting scheduler performance. They broadly discussed these factors, including challenges related to resource utilization, and total execution time. Additionally, they categorized existing scheduling approaches into adaptive, constrained, dynamic, and multi-objective groups, summarizing their advantages and disadvantages. Furthermore, they classified scheduling issues into categories like resource management, data management (including data locality, placement, and replication), fairness, workload balancing, and fault-tolerance, and analyzed approaches to address these issues, grouping them into four main categories: dynamic scheduling, constrained scheduling, adaptive scheduling, and others. However, there is no comparison between these studies' advantages and disadvantages, and there is no information about the environment and the platform for implementation or simulation of the surveyed articles.

Khezr et al. [27] proposed a comprehensive review of the applications, challenges, and architecture of MapReduce. This review aims to highlight the advantages and disadvantages of various MapReduce implementations in order to discuss the differences between them. They examined the utilization of MapReduce in multi-core systems, cloud computing, and parallel computing environments. This study provides a comprehensive review of MapReduce applications. Additionally, open issues and future work are discussed. However, the survey is not systematic and did not outline a clear process for selecting articles. Moreover, it did not specifically focus on Hadoop.

Senthilkumar et al. [28] explored different scheduling issues, such as locality, fairness, performance, throughput, and load balancing. They proposed several job scheduling algorithms, including FIFO scheduler, Fair scheduler, Delay scheduler, and Capacity scheduler, and evaluated the pros and cons of each algorithm. The study also discussed various tools for node allocation, load balancing, and job optimization, providing a comparative analysis of their strengths and weaknesses. Additionally, optimization techniques to maximize resource utilization within time and memory constraints were reviewed and compared for their effectiveness. However, the survey is not systematic and did not outline a clear process for selecting articles. Furthermore, there is no comparison between these studies' advantages and disadvantages, and there is no information about the environment and the platform for implementation or simulation of the surveyed articles.

Hashem et al. [29] presented a comprehensive review of MapReduce challenges, including data access, data transfer, iteration, data processing, and declarative interfaces. Bibliometric analysis and review are performed on MapReduce research assessment publications indexed in Scopus. Also, the MapReduce application is included from 2006 to 2015. Furthermore, it discusses the most significant studies on MapReduce improvements and future research directions related to big data processing with MapReduce. It is suggested that power management in Hadoop clusters on MapReduce is one of the main issues to be addressed. In this study, the paper selection process is clear, but there is no comparison between these studies' advantages and disadvantages, and there is no information about the environment and the platform for implementation or simulation of the surveyed articles.

Li et al. [30] addressed the basic concept of the MapReduce framework, its limitations, and the proposed optimization methods. These optimization methods are categorized into several subjects, including job scheduling optimization, improvement in the MapReduce programming model, support for stream data in real-time, performance tuning such as configuration parameters, energy savings as a major cost, and enhanced authentication and authorization. However, there is no comparison between these studies' pitfalls and advantages, and there is no information about the environment and the platform for implementation or simulation of the surveyed articles. Moreover, the survey is not systematic, and the process of selecting articles is not clear.

Tiwari et al. [31] introduced a multidimensional classification framework for comparing and contrasting various MapReduce scheduling algorithms. The framework was based on three dimensions: quality attribute, entity scheduled, and adaptability to runtime environment. The study provided an extensive survey of scheduling algorithms tailored for different quality attributes, analyzing commonalities, gaps, and potential improvements. Additionally, the authors explored the trade-offs that scheduling algorithms must make to meet specific quality requirements. They also proposed an empirical evaluation framework for MapReduce scheduling algorithms and summarized the extent of empirical evaluations conducted against this framework to assess their thoroughness. However, the survey is not systematic and did not outline a clear process for selecting articles. Furthermore, there is no comparison between these studies' advantages and disadvantages, and there is no information about the environment and the platform for implementation or simulation of the surveyed articles.

Polato et al. [32] performed a systematic literature review to establish a taxonomy for classifying research

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 8 of 30

related to the Hadoop framework architecture. They categorized the studies into four main categories: MapReduce, Data Storage & Manipulation, Hadoop Ecosystem, and Miscellaneous. The MapReduce category encompassed studies on solutions utilizing the paradigm and associated concepts, while the Data Storage & Manipulation category covered research on HDFS, storage, replication, indexing, random access, queries, DBMS infrastructure, Cloud Computing, and Cloud Storage. The Hadoop Ecosystem category focused on studies exploring new approaches within the Hadoop Ecosystem, and the Miscellaneous category included research on topics like GPGPU, cluster cost management, data security, and cryptography. The taxonomy developed in this study provides a comprehensive overview of the diverse research landscape surrounding the Hadoop framework architecture. In this study, the paper selection process is clear, but there is no comparison between these studies' advantages and disadvantages, and there is no information about the environment and the platform for implementation or simulation of the surveyed articles.

We compared and highlighted shortcomings of these surveys. Table 1 provides a summary of them and their main properties. The analysis table includes references, key ideas, systematic survey, advantages and disadvantages, comparison algorithms, and evaluation techniques.

Schedulers in Hadoop MapReduce

To respond to RQ2, RQ3, and RQ4, a thorough review of the selected studies was conducted and the most frequently addressed scheduling issues in Hadoop MapReduce were analyzed. We classified the schedulers into six categories:

- Deadline-aware schedulers;
- Data Locality-aware schedulers;
- Cost-aware schedulers;
- Resource-aware schedulers;
- Makespan-aware schedulers;
- Learning-aware Schedulers.

The idea of each paper has been validated by comparing the performance against existing solutions and benchmarks, which is shown in Tables 2, 3, 4, 5, 6, 7 and 8 in "Deadline-aware Schedulers" to "Learning-aware schedulers" sections, respectively. Each category will be discussed in the following subsections.

Deadline-aware schedulers

Some MapReduce jobs on big data platforms have deadlines and need to be completed within those deadlines. When a job has a deadline, the proper resources must be allocated to the job; otherwise, the deadline cannot be satisfied [87]. Therefore, meeting the job deadline is crucial in MapReduce clusters. We classify deadline-aware schedulers into two categories: deadline-aware schedulers in heterogeneous Hadoop Clusters and deadline-aware schedulers in homogeneous Hadoop clusters. In this section, we first survey and review the most popular deadline-aware schedulers, which minimize job deadline misses. Finally, the reviewed schedulers are compared and summarized.

Deadline-aware schedulers in homogeneous clusters

Gao et al. [33] proposed a deadline-aware preemptive job scheduling strategy, called DAPS, for minimizing job deadline misses in Hadoop Yarn clusters. The proposed method considers the deadline constraints of MapReduce applications and maximizes the number of jobs that meet their deadlines while improving cluster resource utilization. DAPS is formulated as an online optimization problem, and a preemptive resource allocation algorithm is developed to search for a good job scheduling policy. DAPS is a distributed scheduling framework that includes a central resource scheduler, a job analyzer, and node resource schedulers.

Cheng et al. [34] provided a MapReduce job scheduler for deadline constraints called RDS. This scheduler assigns resources to jobs based on resource prediction and job completion time estimation. The problem of job scheduling was modeled as an optimization problem. To find the optimal solution, a receding horizon control algorithm was used. They estimated job completion times using a self-learning model.

Kao et al. [35] proposed a scheduling framework, called DamRT, to provide deadline guarantees while considering data locality for MapReduce jobs in homogeneous systems. In the proposed method, tasks are non-preemptive. DamRT schedules the jobs in four steps: Firstly, DamRT determines the dispatching order of jobs. The urgency value instead of the deadline determines the dispatching order of jobs if the map tasks of jobs can be distributed across nodes concurrently. In contrast, if the map tasks of jobs can be allocated across nodes concurrently, the urgency value determines the priority of the jobs. An "urgency value" is calculated by dividing the estimated response time by the slack time slots across nodes. DamRT first dispatches the job with the highest urgency value. Secondly, the partition order is adjusted for all map tasks of the job. The scheduler assigns the map tasks of a job across nodes, based on the access probability of the required data of the nodes. Thirdly, DamRT assigns the map tasks to nodes according to data locality and blocking time. If two map tasks are assigned to one node, one task will wait for another task's data transfer and execution. For other tasks, the blocking time is defined by the

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 9 of 30

Table 1 Surveys and their properties

Reference	Year	Key Ideas	Systematic Survey	The process of selecting surveys	Advantages/ Disadvantages	Scheduling Metrics	Comparison Algorithms	Evaluation Techniques
Ghazali et al. [23]	2021	A classification of Hadoop job schedulers based on performance optimization approaches	No	Is not clear	Considering	Considering	Not Considering	Considering
Abdallat et al. [24]	2019	Hadoop MapReduce job scheduling algorithms survey and use cases	No	Is not clear	Not considering	Considering	Not Considering	Not Considering
Hashem et al. [25]	2018	MapReduce scheduling algorithms: a review	No	Is not clear	Considering	Not considering	Not considering	Not considering
Soualhia et al. [26]	2017	Task Scheduling in Big Data Platforms: A Systematic Literature Review	Yes	Is clear	Not considering	Not considering	Not considering	Not considering
Khezr et al. [27]	2017	MapReduce and its applications, challenges, and architecture: a comprehensive review and directions for future research	No	Is not clear	Not considering	Not considering	Not considering	Not considering
Senthilkumar et al. [28]	2016	A Survey on Job Scheduling in Big Data	No	Is not clear	Not considering	Not considering	Not considering	Not considering
Hashem et al. [29]	2016	MapReduce: review and open challenges	No	Is clear	Not considering	Not considering	Not considering	Not considering
Li et al. [30]	2016	MapReduce parallel programming model: a state-of-the-art survey	No	Is not clear	Not considering	Not considering	Not considering	Not considering
Tiwari et al. [31]	2015	Classification Framework of MapReduce Scheduling Algorithms	No	Is not clear	Not considering	Not considering	Not considering	Not considering
Polato et al. [32]	2014	A comprehensive view of Hadoop research—A systematic	Yes	Is clear	Not considering	Not considering	Not considering	Not considering

partition value. Also, the scheduler calculates the partition value of all the nodes for all the map tasks of the job and assigns these map tasks to the node that can schedule the job and has the smallest partition value. Finally, after completing all the map tasks of the job, the reduce tasks of the job are ready for execution. The urgency value instead of the deadline determines the priority of the reduce tasks, if the reduce tasks can be distributed across nodes concurrently. Then, all reduced tasks are allocated to the node with the lowest load, where this job can be scheduled.

Verma et al. [36] extended ARIA and proposed a deadline-based Hadoop scheduler called MinEDF-WC (minimum Earliest Deadline First-Work conserving). They integrated three mechanisms: 1) a policy for job ordering in the queue: when the job profile is

Table 2 Deadline-aware schedulers in homogeneous clusters and their properties

Reference	Year	Key Ideas	Advantages	Disadvantages	Comparison Algorithms	Evaluation Techniques
Gao et al. [33]	2022	Deadline aware preemptive job scheduling in Hadoop Yarn clusters	Minimizing the job deadline misses. Improving the resource utilization	Not considering heterogeneous clusters. No ensuring the deadline of all jobs was met	Capacity Scheduler, EDF, PDSon-Queue (PDQ)	Implementation: Hadoop YARN cluster with five nodes
Cheng et al. [34]	2018	RDS: deadline-aware MapReduce job scheduling with dynamic resource availability	Reducing job deadline misses Reducing job completion time Predicting future resource availability Using flexible deadline time	Not considering heterogeneity Not considering data locality	Fair, Earliest Deadline First (EDF)	Implementation: on a Hadoop cluster
Kao et al. [35]	2016	DamRT: Using dispatcher and schedulability test for deadline guarantees	Increasing the number of jobs that meet the deadlines Increasing data locality Minimizing response time of tasks	Using a static manner to divide the job deadline into task deadlines Not considering heterogeneity	FIFO, EDF, MTSD	Simulation: CloudSimRT simulator
Verma et al. [36]	2012	Using EDF policy for job ordering in the processing queue Using MinEDF and MinEDF-WC mechanisms for allocating a tailored number of Map and reduce slots to each job	Increasing resource utilization Reducing job completion time Minimizing the number of deadline-over jobs	Not considering the heterogeneity	EDF, MinEDF, MinEDF-WC	Implementation: Hadoop cluster with 66 nodes Simulation: SimMR simulation environment
Phan et al. [37]	2011	Using EDF/MR policy to minimize miss rate Using EDF/TD policy to minimize total tardiness	Minimizing deadline miss ratio and tardiness Considering data locality	Not considering the heterogeneity	FIFO, Fair, Capacity	Implementation: Hadoop cluster (21 nodes) on Amazon EC2
Kc et al. [38]	2014	Using a schedulability test to determine whether the job can be completed within the specified deadline or not	Maximizing the number of jobs that can be run while satisfying the deadlines Increasing resource utilization	Nodes are homogeneous data distribution is uniform	No comparison, different input datasets in different possible scenarios	Implementation: in virtualized cluster and physical cluster
Teng et al. [39]	2014	Pausing between Map and reduce stages Using a schedulability bound to test the schedulability of real-time tasks	Meeting the job deadlines Increasing cluster utilization Providing deadline guarantees for the jobs	Not considering heterogeneity Not considering dependent tasks, periodic tasks, non-preemptive execution	FIFO, RM	Implementation: on the Hadoop testbed Implementation: on Hadoop 1.1.0 Simulation: SimMapReduce simulation environment
Wang et al. [40]	2015	SAMES: Using a scheduling algorithm based on the most effective sequence (SAMES) for deadline-constraint jobs	Completing the jobs before their deadlines Increasing system utilization	Not considering heterogeneity Not considering the differences between the Map and Reduce tasks	DC, MinEDF-WC	Implementation: on Hadoop (1.0.4)

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 11 of 30

Table 2 (continued)

Reference	Year	Key Ideas	Advantages	Disadvantages	Comparison Algorithms	Evaluation Techniques
Dong et al. [41]	2011	Using a sampling-based approach, and resource allocation model to design a deadline scheduler, for real-time jobs. Using two-level scheduler to schedule mixed real-time and non-real-time jobs	Increasing system utilization Assigning minimum number of resources to real-time jobs Providing deadline guarantee for the real-time jobs	No discussing the hardware heterogeneity Using a static manner to divide the job deadline into task deadlines	Polo's model, Kc's model	Implementation: on a eleven-node cluster configured with the hadoop-0.20.2 release
Verma et al. [42]	2011 b	Using scaling technique and performance models for estimate the resources required while meet SLOs	Estimating required resources to meet job deadlines Increasing system utilization	Not considering heterogeneity	No comparison, different input datasets in different possible scenarios	Implementation: on a Hadoop cluster

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 12 of 30

Table 3 Deadline-aware schedulers in heterogeneous clusters and their properties

Reference	Year	Key Ideas	Advantages	Disadvantages	Comparison Algorithms	Evaluation Techniques
Jabbari et al. [43]	2021	A cost-efficient resource provisioning and scheduling approach for deadline-sensitive MapReduce computations in cloud environment	Guaranteeing the deadline. Reducing the total hiring cost	Not considering data locality. Not considering data distribution	No comparison, different input datasets in different possible scenarios	Simulation
Shao et al. [44]	2018	EFS: efficient jobs scheduling approach for big data applications	Meeting the job deadline	Not considering data distribution and replication dependencies	AlwaysOn, OPT, AutoScale	Simulation: using Scheduler Load Simulator (SLS)
Lin et al. [45]	2019	DGIA: deadline-constrained and influence-aware design for allocating MapReduce jobs in cloud computing systems	Meeting the job deadline Considering the performance influence over existing tasks	Not considering data locality	O-Hadoop, OR-Hadoop, BGMRS, SDHP, EDFLWF	Simulation: 1000 nodes
Chen et al. [46]	2015	DCMRS: Using the bipartite graph modelling to obtain the optimal scheduling solution	Reducing job execution time Dynamically adjust the task deadlines The low proportion of jobs miss their deadline	No improving data locality No ensuring the deadline of all jobs was met High computational time	ORP, AUMD, ADAPT, MDF, MLF	Implementation: Hadoop Cluster with 24 nodes (4 PMs on Hadoop cluster, 20 VMs on Cloud) Simulation: on the MatLab
Tang et al. [47]	2013	MTSD: Using a node classification algorithm to classify the nodes according to processing capacity	Meeting the deadline constraints Increasing map task's data locality Reducing completion time Improving the precision of task's remaining time evaluation	No solving the reduce task scheduling problem No providing deadline guarantees for the jobs. Using a static manner to divide the job deadline into task deadlines	Fair, FIFO	Implementation: in version of Hadoop-0.21
Verma et al. [48]	2011 a	ARIA: Using the job profile to estimate the job completion time and the number of resources required for job completion within the deadline	Automatically allocate the resources to the job for meeting the deadline Increasing resource utilization Considering heterogeneous environments	Not considering Map deadline and reduce deadline Not considering node failures	No comparison, different input datasets in different possible scenarios	Implementation: on Hadoop cluster using Hadoop 0.20.2 Simulation: a discrete event simulator
Polo et al. [49]	2013	Estimating the completion time of jobs, based on the average task length of the completed tasks	Dynamically allocate resources to the jobs for meeting their deadline Increasing system throughput Maximizing data locality Considering hardware heterogeneity	Does not interrupt tasks that are already executing Not considering the differences between the Map and reduce tasks	Fair, Basic Adaptive Scheduler	Implementation: on Hadoop 0.21

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 13 of 30

Table 4 Data locality-aware schedulers and their properties

Reference	Year	Key Ideas	Advantages	Disadvantages	Comparison Algorithms	Evaluation Techniques
Kalia et al. [50]	2022	Improving MapReduce heterogeneous performance using KNN fair share scheduling	Improving the data locality. Reducing the execution time. Improving the throughput	No pipelining between Map and reduce phases	Default Hadoop scheduler	Implementation: Hadoop cluster with 20 nodes
Li et al. [51]	2022	Performance optimization of computing task scheduling based on the Hadoop big data platform	Improving the data locality. Reducing the execution time. Improving the performance	Not considering node failures	Default Hadoop scheduler	Implementation: Hadoop cluster with six nodes
Fu et al. [52]	2020	OptLTS: optimal locality-aware task scheduling algorithm based on bipartite graph modelling for Spark applications	Improving the data locality Decreasing job execution time Reducing the network traffic and access latency	Not considering heterogeneity	DefMapTS, MaxNLT, DefRedTS, MinCRT	Implementation: cluster with 8 nodes
Gandomi et al. [53]	2019	HybSMRP: a hybrid scheduling algorithm in Hadoop MapReduce framework	Increasing data locality Decreasing job completion time Avoiding wasting resources	Not considering heterogeneity	Fair, FIFO	Implementation: Hadoop cluster with 21 nodes
He et al. [54]	2011	each node is given an equal chance of getting its local tasks before any non-local tasks are allocated to any node	Achieving high data locality Reducing response time for Map tasks High cluster utilization	No particular	FIFO, Delay	Implementation: Hadoop cluster (private cluster) with 31 nodes
Ibrahim et al. [55]	2012	Maestro: replica-aware Map scheduling for MapReduce	Improving locality of Map tasks Reducing execution time of Map tasks Reducing speculative Map tasks	Lacks the ability to work in a dynamic environment on the public cloud	Native Hadoop	Implementation: on a local virtualized testbed and Grid'5000 test bed (100 nodes)
Zhang et al. [56]	2011 a	Using a data locality-aware scheduling method	Improving data locality Improving job's execution time Reducing job's response time	Task reservation incur on runtime overhead, e.g., waiting time Not considering homogeneous environment	Default Hadoop	Implementation: in Hadoop-0.20.2
Zhang et al. [57]	2011 b	Using the next-k-node scheduling (NKS) method to calculate the probability of each Map task, and schedule the one with the highest probability	Improving data locality of Map tasks Reducing execution time Reducing network load	Supposing all nodes process Map tasks at the same rate in homogeneous environment In heterogeneous environments, such an assumption cannot be made	Default Hadoop	Implementation: on hadoop-0.20.2. in a cluster including ten nodes
Zaharia et al. [58]	2010	Delay: an effective data locality-aware task scheduling method for MapReduce framework in heterogeneous environments	Achieving high data locality Simplicity of scheduling Improving response times for small jobs Increasing throughput Avoiding job starvation	Ineffective if many tasks are longer than average jobs or if nodes have few slots Allowing a node to obtain multiple non-local Map tasks in a heartbeat interval if the node has more than one free slot	FIFO, Fair, Fair with delay scheduling	Implementation: in two environments: Amazon EC2 and a 100-node private cluster

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 14 of 30

Table 5 Cost-aware schedulers and their properties

Reference	Year	Key Ideas	Advantages	Disadvantages	Comparison Algorithms	Evaluation Techniques
Seethalakshmi et al. [59]	2022	Real-coded multi-objective genetic algorithm with effective queuing model for efficient job scheduling in heterogeneous Hadoop environment	Reducing the cost. Minimizing the execution time. Improving the resource utilization	No particular	GA, FIFO, fruit fly, firefly	Simulation: MRSIM simulator
Tang et al. [60]	2021	Cost-efficient workflow scheduling algorithm for applications with deadline constraint on heterogeneous Clouds	Reducing the total workflow execution cost	No ensuring the deadline of all jobs was met	LHCM, PCP	Simulation: CloudSim
Vinutha et al. [61]	2021	Budget Constraint Scheduler for Big Data Using Hadoop MapReduce	Reducing the budget. Reducing the completion time	Not considering node failures. Not considering data locality	FIFO, Fair scheduler	Implementation: Hadoop cluster with 12 data nodes
Javanmardi et al. [62]	2020	A unit-based, cost-efficient scheduler for heterogeneous Hadoop systems	Reducing the cost of executing jobs. Minimizing the jobs execution times. Improving data locality	No pipelining between Map and reduce phases	FIFO, Fair	Implementation: Hadoop cluster with 6 nodes Simulation: MRSIM simulator
Rashmi et al. [63]	2016	Deadline constrained cost effective workflow scheduler for Hadoop clusters in cloud datacenter	Reducing the cost. Meeting the deadline	They have considered only VM cost, but there are other costs in terms of bandwidth usage, data store usage	SciCumulus adaptive approach	Implementation: cloudsim environment
Zacheilas et al. [64]	2016	ChEsS cost-effective scheduling across multiple heterogeneous Mapreduce clusters	Minimizing the cost. Optimizing the execution time	Not considering data locality	NSGA-II, GDE3, Weighted Sum, Starfish	Simulation
Palanisamy et al. [65]	2015	Cost-effective resource provisioning for MapReduce in a cloud	Reducing the cloud compute infrastructure cost. Reducing job response times	No ensuring the deadline of all jobs was met. Not considering the complex workloads	Dedicated cluster, Per-job Cluster	Simulation
Chen et al. [66]	2014	CRESP: towards optimal resource provisioning for MapReduce computing in public clouds	Minimizing the monetary or time cost	Not considering heterogeneity	No comparison, different input datasets in different possible scenarios	Implementation: 16-node Hadoop cluster and Amazon EC2

Table 6 Resource-aware schedulers and their properties

Reference	Year	Key Ideas	Advantages	Disadvantages	Comparison Algorithms	Evaluation Techniques
Aarthee et al. [67]	2023	Heuristic scheduling using bin packing MapReduce scheduler for heterogeneous workloads performance in big data	Increasing the resource utilization. Improving the makespan. The number of data nonlocal execution is low	Not considering data distribution. Not considering node failures	RWS, HMJS	Implementation: Hadoop cluster with twelve nodes
Jeyaraj et al. [68]	2022	Optimizing MapReduce Task Scheduling on Virtualized Heterogeneous Environments Using Ant Colony Optimization	Improving resource utilization. Minimizing the makespan	No implementation	FS, RWS, HMJS	Simulation
Zhang et al. [69]	2015	PRISM: a phase-level resource-aware MapReduce scheduler	Improving resource utilization Reducing job running time Avoiding resource contention	Worse data locality Not considering heterogeneity	Fair, Yarn	Implementation: in Hadoop 0.20.2
Rasooli et al. [70]	2014	Using an algorithm which classifies the jobs based on their requirements and finds an appropriate matching of resources and jobs in the system	Reducing average completion time Improving fairness Satisfying the required minimum shares Increasing locality No starvation for small job Avoiding the sticky slot problem	Increasing scheduling overhead No separating data intensive and computation intensive jobs in performing the classification	FIFO, Fair	Implementation: Hadoop cluster with four nodes Simulation: MRSIM, a MapReduce simulator
Polo et al. [71]	2011	Using job profiling information to dynamically adjust the number of job slots and their placement	Improving resource utilization Meeting completion time goals Reducing makespan	Not considering heterogeneity The task's resource consumption is assumed to be stable during its lifetime No support job priority	Fair	Implementation: on a Hadoop cluster
Sharma et al. [72]	2012	MROrchestrator: Using fine-grained, dynamic, and coordinated resource allocation instead of slot-based resource allocation	Increasing resource utilization Reducing job completion time Dynamically identify resource bottlenecks, and resolve them	Not considering other resources like disk and network bandwidth No evaluating on a large cloud Neglecting the workload imbalance among tasks Not considering heterogeneity	Mesos, NGM	Implementation: on two 24-node physical and virtualized Hadoop clusters
Pastorelli et al. [73]	2015	Using size-based scheduling with aging	Decreasing system response times Guarantying Fairness Avoiding job starvation Considering data locality	No implementation in large clusters	Fair	Implementation: on a cluster composed of 20 Task Tracker worker machines
Tian et al. [74]	2011	Using cost model to find the optimal resource provisioning for different decision problems	Cost model fits well on four tested programs Cost model has low error rates	Not conducting experiments in the public cloud Not considering heterogeneity	No comparison, different input datasets in different possible scenarios	Implementation: in their in-house 16-node Hadoop cluster

Hedayati *et al. Journal of Cloud Computing* (2023) 12:143

Page 16 of 30

Table 6 (continued)

Reference	Year	Key Ideas	Advantages	Disadvantages	Comparison Algorithms	Evaluation Techniques
Ghoneem et al. [75]	2017	Providing the classifier with information about jobs requirement and node capabilities	Increasing resources utilization Minimizing average completion time Minimizing master node overhead No starvation for small job Avoiding the sticky slot problem Considering heterogeneity	Finding content information is computationally expensive and time consuming	HP model, Starfish	Implementation: on a cluster consisted of three nodes

Table 7 Makespan-aware schedulers and their properties

Reference	Year	Key Ideas	Advantages	Disadvantages	Comparison Algorithms	Evaluation Techniques
Varalakshmi et al. [76]	2022	Optimized scheduling of multi-user Map-Reduce jobs in heterogeneous environment	Reducing the makespan, Improving resource utilization, near-zero wait time	Not considering data locality. Not considering adaptable dynamic environment	FIFO	Implementation
Maleki et al. [77]	2021	SPO: A Secure and Performance-aware Optimization for MapReduce Scheduling	Minimizing makespan Maximizing security with risk-limitation constraint Traffic reduction	No particular	FIFO, HCS	Simulation
Maleki et al. [78]	2020	TMaR: a two-stage MapReduce scheduler for heterogeneous environments	Improving the power consumption of cluster Minimizing the makespan of a batch of tasks Reducing the network traffic Considering heterogeneous environments	No particular	Hadoop-stock, Hadoop-A	Simulation
Jiang et al. [79]	2017	Minimizing Makespan for MapReduce systems with different servers	Minimizing the makespan of servers	Not considering Heterogeneity of resources and jobs	FIFO, Fair, LPT	Simulation: cluster with 50 nodes
Verma et al. [80]	2013	Using a heuristic, called BalancedPools to optimize the jobs execution	Improving makespan Increasing resource utilization	Not considering jobs with dependency, e.g., MapReduce workflow Not considering heterogeneity	Min strategy, Max strategy, MinSim, MaxSim	Implementation: on Hadoop cluster. using Hadoop 0.20.2 Simulation: SimMR
Yao et al. [81]	2015	Using slot ratio between Map and Reduce tasks as a tunable knob to improve the makespan Using Workload Monitor and Slot Assigner to automate the slot assignment ratio	Minimizing makespan Increasing resource utilizations Dynamically allocates slots to tasks Considering the heterogeneity	Not considering data locality	FIFO	Implementation: in Hadoop V0.20.2
Zheng et al. [82]	2016	Proposing joint scheduling optimization of overlapping Map and shuffle phases	Minimizing makespan Reducing average job execution time Increasing resource utilization	Leading to an overly large job waiting time	MaxDiff, Pairwise, MaxShuffle, MaxSRPT	Conducting real data-driven experiments
Tang et al. [83]	2016	Using an optimized MapReduce workflow scheduling algorithm	Minimizing makespan Low schedule length Increasing parallel speedup Higher efficiency Considering data locality	No particular	MRWS_NPI, SWS, WS_NWH	Implementation: in a Heterogeneous cluster

Table 8 Deadline-aware schedulers and their properties

Reference	Year	Key Ideas	Advantages	Disadvantages	Comparison Algorithms	Evaluation Techniques
Ghazali et al. [84]	2022	CLQLMRS: improving cache locality in MapReduce Job scheduling using Q-learning	Reducing execution time. Improving the data locality. Improving Hadoop performance	Need to train the scheduling policy, which may be challenging if environmental changes occur rapidly and retraining becomes difficult	FIFO, adaptive cache local scheduling (ACL)	Implementation: Cluster with 8 nodes
Naik et al. [85]	2017	A learning-based MapReduce scheduler in heterogeneous environments	Improving the data locality. Improving the performance of MapReduce	Not considering job deadline and data locality	Not considering	Not considering
Naik et al. [86]	2015	Performance Improvement of MapReduce framework in heterogeneous context using reinforcement learning	Minimizing job completion time Does not require prior knowledge of environmental characteristics	Not considering job deadline and data locality	Not considering	Not considering

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 19 of 30

unavailable, job ordering is calculated by the EDF policy (Earliest Deadline First). The EDF policy is used with Hadoop's default scheduler, which allocates the maximum number of slots to job. When a new job that has an earlier deadline arrives, the active tasks are killed in this scheme. 2) a mechanism for allocating an appropriate number of map and reduce slots required for the job to satisfy the deadline is proposed to improve the EDF job ordering. This mechanism assigns the minimum resource required to satisfy a job deadline when the job profile is available. As a result, it is called MinEDF. 3) a mechanism for allocating and deallocating extra resources among the jobs: the authors presented a mechanism that improves the MinEDF. It is called minEDF-WC and is designed to use spare slots efficiently. After allocating the minimum number of slots required for the job, the unallocated slots are referred to as spare slots. These spare slots are allocated to active jobs. When the new job with an earlier deadline arrives, the mechanism determines whether it can meet its deadline after completing the running tasks and their slots are released. If released slots cannot meet the new job's deadline, the mechanism stops the active tasks and reallocates these slots to the new job.

Phan et al. [37] showed that the default schedulers in Hadoop are inadequate for deadline-constrained MapReduce job scheduling. They presented two deadline-based schedulers that are based on the Earliest Deadline First (EDF) but customized for Hadoop environment. The first is EDF/TD, which minimizes total or maximum tardiness. A job's tardiness is the elapsed time between its deadline and completion time. The second is EDF/MR, which minimizes the miss rate. The "miss rate" is the number of soft real-time jobs that miss their deadlines. The EDF/TD scheduling policy sorts the job queue based on job deadlines. For each job, if it has local tasks on a node, one of those is selected to be scheduled according to the lowest value of laxity (difference between the deadline and the estimated execution time). If all the tasks of the job are remote tasks, only the tasks whose data is close to the node are executed. If there are no such tasks, these steps are repeated for the following job in the queue by the scheduler. Finally, In the absence of any tasks, the first task of the first job in the queue is chosen. The EDF/MR scheduling policy classifies the jobs into two sets: schedulable jobs are those that are expected to meet their deadlines, and unschedulable jobs are those that are not predicted to meet their deadlines. A job is predicted to meet its deadline if the present time plus the estimated remaining execution time is less than the job's deadline. The scheduler first considers the schedulable set for scheduling. When it is empty, the scheduler considers it an unschedulable set. In each set, the priority of the tasks is assigned based on the EDF/TD policy, and tasks with the highest priority are allocated first.

Kc et al. [38] proposed a deadline-based scheduler in Hadoop clusters. To determine if the job can be completed by the deadline, the scheduler first performs a schedulability test. Therefore, it estimates the minimum number of map and reduce slots required for the job to be completed before the deadline. In order to schedule a job, there must be more free slots than or equal to the minimum number for map and reduce [88, 89]. Hence, jobs are only scheduled if they can be finished before the deadline.

Teng et al. [39] addressed the problem of deadline-based scheduling from two perspectives. 1. The real-time scheduling problem is formulated by determining the features of a task, cluster, and algorithm. The task is modeled as a periodic sequence of instances, and the MapReduce cluster is modeled as an exclusive cluster. Also, tasks can be run only sequentially rather than concurrently. Moreover, tasks are run preemptively since the cluster supports preemption. 2. To schedule tasks deadline, they proposed the Paused Rate Monotonic (PRM) method. The highest priority is assigned to the task with the shortest deadline. As mentioned, a task is a periodic sequence of successive instances. As a result, the current instance needs to be finished before the new instance arrives. In this algorithm, the period is the deadline for any instance, thus, the highest priority is given to the task with the lowest deadline. Since a MapReduce job comprises a map task and a reduce task, the authors segmented the period T into a mapping period TM and a reducing period TR. TM is the deadline for map tasks, and TR is the deadline for reduce tasks. PRM pauses the reduce tasks until the map tasks are completed. Thus, the reduce tasks are scheduled only after time TM. By pausing between the map and reduce stages, the resources are utilized efficiently.

Wang et al. [40] presented a scheduling algorithm using the most effective sequence (SAMES) for scheduling jobs with deadlines. First, they introduced the concept of a sequence: an ordered list of jobs. Sequence restricts the order in which the map phase of jobs is finished. Next, they defined the concept of effective sequence (ES): The sequence seq of a job set is an ES if the completion time of each job is shorter than its deadline. They presented two efficient techniques for finding ESes. If there is more than one ES, they utilized a ranking method to select the most effective sequence (MES). An incremental method is proposed for determining whether a new arrival job is acceptable and updating the MES.

Dong et al. [41] addressed the problem of scheduling mixed real-time and non-real-time MapReduce jobs. They proposed a two-stage scheduler that is implemented

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 20 of 30

using multiple techniques. First, by using a sampling-based method called TFS, the scheduler predicts the map and reduce task execution time. Next, each job is adaptively controlled by a resource allocation model (AUMD) to run with a minimum slot. Finally, a two-stage scheduler for scheduling real-time and non-real-time jobs is proposed, which supports resource preemption.

Verma et al. [42] extended ARIA and proposed a novel framework and technique to automate the process of estimating required resources to meet application performance goals and complete data processing by a certain time deadline. The approach involves building a job profile from the job's past executions or by executing the application on a smaller data set using an automated profiling tool. To explain more, they benefited from linear regression to predict the job completion time depending on the size of the input dataset and assigned resources. Scaling rules combined with a fast and efficient capacity planning model are applied to generate a set of resource provisioning options.

We investigated and analyzed deadline-aware schedulers. Table 2 provides a summary of popular deadline-aware schedulers in homogeneous Hadoop clusters and their main properties. The analysis table includes references, key ideas, advantages and disadvantages, comparison algorithms, and evaluation techniques.

Deadline-aware schedulers in heterogeneous clusters

Jabbari et al. [43] addressed the challenge of selecting appropriate virtual machines (VMs) and distributing workload efficiently across them to meet both deadline and cost minimization goals in cloud environments. The paper proposes a cost minimization approach to calculate the total hiring cost before and during the computations, based on the application's input size and the required type and number of VMs. The proposed approach uses a daily price fluctuation timetable to schedule MapReduce computations and minimize the total cost while meeting the deadline.

Shao et al. [44] investigated the service level agreement violation (SLAV) of the YARN cluster using a Fair Scheduling framework. To assign resources to MapReduce jobs, the authors used dynamic capacity management and a deadline-driven policy. A Multi-dimensional Knapsack Problem (MKP) and a greedy algorithm were employed to model and solve the problem, respectively.

Lin et al. [45] provided a deadline-aware scheduler for MapReduce jobs, DGIA, in a heterogeneous environment. Using the data locality, DGIA meets the deadlines of new tasks. When the deadline of some new tasks is not met, DGIA re-allocates these tasks. The task re-allocation problem is transformed into a well-known network

graph problem: minimum cost maximum-flow (MCMF) to optimize the computing resource utilization.

Chen et al. [46] addressed the problem of Deadline-Constrained MapReduce Scheduling called DCMRS in heterogeneous environments. Using Bipartite Graph modeling, they presented a new scheduling method named BGMRS. It has three major modules, i.e., deadline partition, bipartite graph modeling, and scheduling problem transformation. First, the BGMRS adaptively determines deadlines for map and reduce task of a job. Secondly, to demonstrate the relationship between tasks and slots, they formed a weighted bipartite graph. Finally, the DCMRS problem is transformed into the minimum weighted bipartite matching (MWBM) problem to achieve the best allocation between tasks and resources. Also, to solve the MWBM problem, they presented a heuristic method with the node group technique.

Tang et al. [47] presented a deadline-based MapReduce job scheduler called MTSD. In the presented scheduling, user can specify a job's deadline. MTSD presents a node classification algorithm that measures the node's computing capacity. The nodes are classified according to their computing capacity in heterogeneous clusters using this algorithm. One purpose of node classification is to demonstrate a new data distribution model to increase the data locality. Another purpose is to increase the accuracy of the evaluations of the task remaining times. To determine its priority, MTSD computes the minimum number of map and reduce slot required for the job.

Verma et al. [48] proposed ARIA, a framework for deadline-based scheduling in Hadoop clusters. It has three major parts. First, a job profile is created for a production job that runs periodically. A job profile shows the characteristics of the job execution during the map, shuffle, sort, and reduce phases. Second, using the job profile, i) the job completion time is estimated according to the assigned map and reduce slots, and ii) the minimum number of map and reduce slots for meeting the job's deadline is estimated based on Lagrange's method. Finally, they benefited from the earliest deadline first policy (EDF) to determine job ordering.

Polo et al. [49] presented the adaptive scheduler for MapReduce multi-job workloads with deadline constraints. The scheduler divides a job into tasks already completed, not yet started, and currently running. It adaptively determines the number of slots required for the job to satisfy the deadline. Therefore, for each job, the amount of pending work is estimated. To this end, this technique investigates both the tasks that have not yet started and the currently running tasks. Based on these two parameters, it calculates the number of slots. Then, the scheduler calculates the priority of each job based on the number of slots to be assigned simultaneously to

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 21 of 30

each job. Jobs are sorted into a queue based on their priority. In order to account for the hardware heterogeneity, nodes can be classified into two groups: those with general-purpose cores and those with specialized accelerators such as the GPU. When a job requires a GPU to run its tasks, the scheduler assigns slots from the nodes with the GPUs to run the tasks of the job [90].

We investigated and analyzed deadline-aware schedulers. Table 3 provides a summary of popular deadline-aware schedulers in heterogeneous Hadoop clusters and their main properties. The analysis table includes references, key ideas, advantages and disadvantages, comparison algorithms, and evaluation techniques.

Data locality-aware schedulers
In data locality-aware schedulers, tasks are allocated to the node where the task's input data is stored; otherwise, they are assigned to the node closest to the data node [56]. Researchers proposed several scheduling algorithms to improve data locality because it minimizes data transfer over the network and mitigates the total execution time of tasks, thus improving the Hadoop performance [4]. Therefore, improving data locality is a crucial problem in MapReduce clusters. In this section, we review several important data locality-aware schedulers.

Kalia et al. [50] tackled the issue of heterogeneous computing nodes in a Hadoop cluster, which can lead to slower job execution times due to varying processing capabilities. To address this challenge, the authors introduced a K-Nearest Neighbor (KNN) based scheduler that employs speculative prefetching and clustering of intermediate map outputs before sending them to the reducer for final processing. The proposed algorithm prefetches input data and schedules intermediate key-value pairs to reduce tasks using the KNN clustering algorithm with Euclidean distance measure. The study concludes that their scheduler, based on clustering, enhances data locality rate and improves execution time.

Li et al. [51] concentrated on optimizing computing task scheduling performance in the Hadoop big data platform. They introduced an enhanced algorithm for task scheduling in Hadoop, which evaluates the data localization saturation of each node and prioritizes nodes with low saturation to prevent preemption by nodes with high saturation. The authors concluded that their proposed scheduler enhances data locality, overall performance, and reduces job execution time in the Hadoop environment.

Fu et al. [52] addressed the problem of cross-node/rack data transfer in the distributed computing framework of Spark, which can lead to performance degradation, such as prolonging of entire execution time and network congestion. The authors propose an optimal locality-aware task scheduling algorithm that utilizes bipartite graph modelling and considers global optimality to generate the optimal scheduling solution for both map tasks and reduce tasks for data locality. The algorithm calculates the communication cost matrix of tasks and formulates an optimal task scheduling scheme to minimize overall communication cost, which is transformed into the minimum weighted bipartite matching problem. The problem is resolved by the Kuhn-Munkres algorithm. The paper also proposes a locality-aware executor allocation strategy to improve data locality further.

Gandomi et al. [53] discussed the importance of data locality-aware scheduling in the Hadoop MapReduce framework, which is designed to process big data on commodity hardware using the divide and conquer approach. The authors propose a new hybrid scheduling algorithm called HybSMRP, which focuses on increasing data locality rate and decreasing completion time by using dynamic priority and localization ID techniques. The algorithm is compared to Hadoop default scheduling algorithms, and experimental results show that Hyb-SMRP can often achieve high data locality rates and low average completion times for map tasks.

He et al. [54] presented a map task scheduler called MatchMaking to increase data locality. First of all, when the first job does not have a local map task to the requesting node, the scheduler searches for the succeeding jobs. Then, each node is given an equal chance of getting its local tasks, to do this, if the node is unable to find a local task for the first time in a row, it will not be allocated any non-local tasks. Therefore, the node does not get a map task during this heartbeat interval. Nodes that fail to find a local task a second time are assigned a non-local task in order to prevent the waste of computing resources. Each slave node is allocated a status marker based on its locality. Based on the marked value of a slave node, the scheduler determines if a non-local task is allocated to the slave node if there is no map task local to it. Third, in this algorithm, a slave node can be assigned a maximum of one non-local task per heartbeat. Lastly, with the addition of a new job to the queue, the locality markers of all slave nodes will be removed. Upon arriving at a new job, the algorithm resets all slave nodes' node statuses since they may have local tasks for a new job.

Ibrahim et al. [55] presented a map task scheduler, Maestro, to increase the data locality. To accomplish this, the chunks' locations, replicas' locations, and how many other chunks each node hosts are tracked by Maestro. There are two waves of scheduling that the Maestro employs: the first wave scheduler and the run-time scheduler. Taking into account the number of map tasks hosted on each node and the replication scheme for the input data, the first wave scheduler fills up empty slots on

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 22 of 30

each node. The run-time scheduler determines a task's probability of being scheduled on a certain machine based on replicas of its input data.

Zhang et al. [56] proposed a MapReduce job scheduling technique to increase the data locality in heterogeneous systems. Upon a node sends a request, the scheduler assigns the task to the input data stored on that node. In the absence of such tasks, the task that has the closest input data to that node is selected and its transmission and waiting times are calculated. The task is reserved for the node storing the input data when the waiting time is shorter than the transmission time.

Zhang et al. [57] provided a map task scheduler, named next-k-node scheduling (NKS), to achieve the data locality. When a requesting node sends a request, the technique schedules tasks that their input data is stored on that node. In the absence of such tasks, a probability is calculated for each map task, and the ones with higher probability are scheduled. A task has a low probability that its input data is stored on the next k nodes, so the technique reserves these tasks for these nodes.

Zaharia et al. [58] proposed a delay scheduling algorithm to increase the data locality. Upon receiving from a requesting node, if the head-of-line job is unable to launch a local task, the scheduler skips and finds a local task from subsequent jobs. The maximum delay time is set to D. When the scheduler skips a job for an extended period of time (D), the job is allocated to the node with non- local data to avoid starvation [91]. We investigated and analyzed the data locality-aware schedulers. Table 4 provides a summary of data locality-aware schedulers and their main properties. The analysis table includes references, key ideas, advantages and disadvantages, comparison algorithms, and evaluation techniques.

Cost-aware schedulers

In big data platforms, data centers store a huge amount of data. Processing this data requires thousands of nodes in Hadoop clusters. Such large clusters consume enormous amounts of power and increase the cost of data centers. Therefore, we face a big challenge in minimizing cost in MapReduce clusters [31]. In this section, we survey and review several important cost-aware schedulers that reduce the cost of MapReduce systems. Finally, the reviewed schedulers are compared and summarized.

Seethalakshmi et al. [59] proposed a new scheduling method based on Real Coded Genetic Algorithm (RCGA) to effectively allocate nodes in heterogeneous Hadoop settings. The paper evaluates metrics such as load, makespan of each Virtual Machine (VM), execution time, and memory constraints of each job to identify the challenges in allocating jobs to nodes. The authors propose a solution based on work classification, where jobs are categorized into 'n' classes based on execution rate, priority, and arrival rate. The best set of work classes for each VM is then proposed to solve the issue of resource and work mismatch. The final scheduling is done by the Real coded GA optimization model, which considers fairness and minimum share satisfaction. The authors conducted experiments, and the results show that it outperforms existing systems in terms of metrics such as execution time, cost, resource utilization, throughput.

Tang et al. [60] addressed the problem of scheduling cloud applications with precedence-constrained tasks that are deadline-constrained and must be executed with minimum financial cost. They proposed a heuristic cost-efficient task scheduling strategy called CETSS, which includes a workflow DDAG model, task sub deadline initialization, greedy workflow scheduling algorithm, and task adjusting method. The proposed greedy workflow scheduling algorithm consists of a dynamical task renting billing period sharing method and an unscheduled task sub deadline relax technique.

Vinutha et al. [61] proposed a scheduling algorithm to optimize the MapReduce jobs for performance improvement in processing big data using Hadoop. The goal of the algorithm is to reduce the budget and execution time of cloud models by establishing a relationship between the scheduling of jobs and the allocation of resources. The earliest finish time is considered for cloud resource optimization to assign the map tasks. The algorithm schedules tasks based on the availability of slots and available resources in the cluster. The authors evaluate their proposed method on word count with different input data sizes.

Javanmardi et al. [62] proposed a high-level architecture model for scheduling in heterogeneous Hadoop clusters. The proposed model reduces the scheduling load by performing part of the scheduling in the user system. They also present a scheduler based on the base unit that can estimate the execution time in heterogeneous Hadoop clusters with low overhead and high accuracy, while being resistant to node failure. The scheduler considers the cost of transfer and processing of jobs in the clusters, which leads to a reduction in the cost of executing the jobs. The paper also designs four algorithms for the scheduler, including the estimation of execution time in the user system, distributing the input data of jobs between data nodes based on performance, job scheduling, and task scheduling.

Rashmi et al. [63] proposed a cost-effective workflow scheduler for Hadoop in cloud data centers. The motivation behind the scheduling issue is the need to efficiently allocate resources to complete MapReduce jobs within the deadline and at a lower cost. The proposed scheduler takes into account the workflow as a whole

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 23 of 30

rather than treating each job separately, as in many existing schedulers. The scheduler creates and maintains virtual machines (VMs) for jobs in a workflow, even after their completion to avoid time wastage and overheads.

Zacheilas et al. [64] proposed a novel framework called ChEsS for cost-effective scheduling of MapReduce workloads in multiple cluster environments. The scheduling problem is challenging due to the tradeoff between performance and cost, the presence of locality constraints, and the use of different intra-job scheduling policies across clusters. The goal of the framework is to automatically suggest jobs-to-clusters assignments that minimize the required budget and optimize the end-to-end execution time of the submitted jobs. The framework estimates the impact of various parameters, such as job locality constraints, on the user's makespan/budget and detects near-optimal job-to-cluster assignments by efficiently searching the solution space.

Palanisamy et al. [65] proposed a new MapReduce cloud service model called Cura for cost-effective provisioning of MapReduce services in a cloud. Unlike existing services, Cura is designed to handle production workloads that have a significant amount of interactive jobs. It leverages MapReduce profiling to automatically create the best cluster configuration for the jobs, implementing a globally efficient resource allocation scheme that significantly reduces the resource usage cost in the cloud. Cura achieves this by effectively multiplexing the available cloud resources among the jobs based on the job requirements and by using core resource management schemes such as cost-aware resource provisioning, VM-aware scheduling, and online virtual machine reconfiguration.

Chen et al. [66] addressed the problem of optimizing resource provisioning for MapReduce programs in the public cloud to minimize the monetary or time cost for a specific job. The authors study the components in MapReduce processing and build a cost function that models the relationship among the amount of data, the available system resources (Map and Reduce slots), and the complexity of the Reduce function for the target MapReduce program. The model parameters can be learned from test runs, and based on this cost model, the authors propose an approach called Cloud RESource Provisioning (CRESP) to solve a number of decision problems, such as the optimal amount of resources that can minimize the monetary cost with the constraint on monetary budget or job finish time.

We investigated and analyzed deadline-aware schedulers. Table 5 provides a summary of popular cost-aware schedulers and their main properties. The analysis table includes references, key ideas, advantages and disadvantages, comparison algorithms, and evaluation techniques.

Resource-aware schedulers

In big data applications, data centers are deployed on large Hadoop clusters. The nodes in these clusters receive a large number of jobs and require more resources to execute them. As a result, race condition arises among the jobs that demand resources like CPU, memory, and I/O [92]. Therefore, improving cluster resource utilization has become a major concern in MapReduce clusters. This section presents some of the most popular resource-aware schedulers which increase resource utilization.

Aarthee et al. [67] proposed a new scheduler, called Dynamic Performance Heuristic-based Bin Packing (DP-HBP) MapReduce scheduler, to improve resource utilization in a heterogeneous virtualized environment. By analyzing the exact combination of vCPU and memory capacities, the scheduler can effectively allocate resources and improve the entire virtual cluster's performance. In other words, the scheduler allocates the proper number of virtual machine cores and memory at the datacentres for cloud users. Also, the scheduler is a generalized model that can handle data-intensive jobs on MapReduce, regardless of their nature.

Jeyaraj et al. [68] addressed the challenge of resource utilization in virtual clusters running Hadoop MapReduce workloads, which can suffer from heterogeneities at the hardware, virtual machine, performance, and workload levels. The authors propose an efficient MapReduce scheduler called ACO-BP that places the right combination of map and reduce tasks in each virtual machine to improve resource utilization. They transform the MapReduce task scheduling problem into a 2-Dimensional bin packing model and obtain an optimal schedule using the ant colony optimization (ACO) algorithm. The ACO-BP scheduler minimizes the makespan for a batch of jobs and outperforms three existing schedulers that work well in a heterogeneous environment. The authors conclude that their proposed scheduler is an effective solution to improve resource utilization in virtual clusters running Hadoop MapReduce workloads.

Zhang et al. [69] presented a phase-level MapReduce scheduler called PRISM. This scheduler divides tasks into phases and schedules tasks at the phase level. PRISM assigns the resources based on the phase that each task is running. The authors proposed a heuristic algorithm to determine the order of the phases that can be scheduled on a machine: Each phase is assigned a utility value, which demonstrates the benefit of scheduling the phase. The utility value is calculated based on the fairness and job performance of the phase. Then the phase that has the highest utility is chosen. Also, the utility value is dependent on the phase. If a phase is a map or shuffle, a new map or reduce task is selected for scheduling. If a phase is a map or shuffle, a new map or reduce task is

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 24 of 30

selected for scheduling. In this case, the scheduler determines the phase's utility by achieving a higher degree of parallelism by performing an extra task. For other phases, PRISM determines the utility of the phase by the urgency of finishing the phase. Urgency is calculated by how long it has been paused in seconds. A task whose execution has been paused for a long time needs to be scheduled as soon as possible.

Rasooli et al. [70] proposed a Hadoop scheduling algorithm called COSHH in heterogeneous environments. Utilizing a k-means clustering algorithm, COSHH categorizes the jobs into classes according to their requirements. Then the scheduler determines which jobs and resources are best suited to each other. To do this, it first builds a linear program (LP) and defines it using the characteristics of the job classes and the resources. After solving the LP, the scheduler finds a class set for each resource. Afterward, COSHH allocates jobs to resources based on fairness and minimum share requirements. Moreover, COSHH is composed of two major steps: first, upon arriving a new job, the algorithm stores it in the proper queue. Second, when a heartbeat message is received, the algorithm allocates a job to a free resource.

Polo et al. [71] proposed the Adaptive Scheduler called RAS for MapReduce multi-job workloads. The main purpose of the method is to utilize the resources efficiently. This scheduler proposes the "job slot" concept instead of the "task slot." Each job slot is a specific slot for a certain job. RAS computes the number of concurrent tasks that need to be assigned to complete a job before its deadline. This calculation is performed using the deadline, the number of pending tasks, and the average task length. Then, each job is assigned a utility value by RAS. The placement algorithm uses a utility value to choose the best placement of tasks on TaskTrackers.

Sharma et al. [72] proposed MROrchestrator, a fine-grained, dynamic, and coordinated resource management framework that effectively manages the resources. Resource bottleneck detection and resource bottleneck mitigation are two functions of the MROrchestrator. First, it collects the run-time resource allocation information of each task and identifies resource bottlenecks. The latter resolves bottlenecks through coordinated resource allocations.

Pastorelli et al. [73] proposed a scheduler for Hadoop called HFSP that achieves fairness and short response times. HFSP utilizes size-based scheduling to assign the cluster resources to the jobs. Job size is required for size-based scheduling, but there is no a priori knowledge of the job size in Hadoop. HFSP estimates job sizes during job execution to construct job size information. Also, using an aging function, the priority of jobs is computed. Afterward, the scheduler allocates resources to jobs based on priority. For both small and large jobs, aging is used to prevent starvation.

Tian et al. [74] discussed the optimal resource provisioning to execute the MapReduce programs in public clouds. Using a cost method, for the target MapReduce job, they modeled the relationship between the amount of input data, Map and Reduce slots, and the complexity of the Reduce function. Using a cost model can figure out how many resources are needed to reduce costs by a specified deadline or to reduce the time under a specified budget [93].

Ghoneem et al. [75] provided a MapReduce scheduling method to improve efficiency and performance in the heterogeneous cluster. The scheduler uses a classification algorithm based on job processing requirements and resources available in order to categorize jobs as executable and nonexecutable. To obtain the best performance, the scheduler allocates the executable jobs to the proper nodes. We described the most popular resource-aware schedulers. Table 6 provides the summary of the main properties of resource-aware schedulers. The analysis table includes references, key ideas, advantages and disadvantages, comparison algorithms, and evaluation techniques.

Makespan-aware schedulers

The makespan (total completion time) of a set of jobs is the total amount of time it takes to complete jobs. In order to increase the cluster's performance, makespan needs to be minimized by distributing the data across the nodes. Also, low makespan is a major factor for any scheduler [94]. Therefore, minimizing the makespan has become an important issue in MapReduce clusters. In this section, we first review several important makespan-aware schedulers. Then, the reviewed schedulers are compared and summarized.

Varalakshmi et al. [76] proposed a new job scheduler, called the virtual job scheduler (VJS), which is designed to schedule MapReduce jobs in a heterogeneous cluster. VJS creates a virtual job set by considering the CPU and IO resource utilization levels of each job waiting in the execution queue. The partitioning algorithm is the core of VJS, and the authors proposed two novel partitioning algorithms: two-level successive partitioning (TLSP) and predictive partitioning (PRED). The goal of TLSP is to optimize the busy time of reducers in environments where the higher-level scheduler is aware of the idle time of individual reducers. On the other hand, the goal of PRED is to optimize the overall time taken by the Reducer phase, including the idle time of reducers, and is found to produce better makespan and near-zero wait time in all scenarios, despite the prediction error

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 25 of 30

associated with it. Therefore, PRED is used to partition the input of individual jobs in the virtual job set.

Maleki et al. [77] presented a secure and performance-aware optimization framework called SPO, to minimize the makespan of tasks using a two-stage scheduler. SPO applies the HEFT algorithm in Map and Reduce stage, respectively, and considers network traffic in the shuffling phase. Moreover, the authors proposed a mathematical optimization model of the scheduler to estimate the system performance while considering the security constraints of tasks.

Maleki et al. [78] proposed a two-stage MapReduce task scheduler for heterogeneous environments called TMaR, which aims to minimize the makespan of a batch of tasks while considering network traffic. The authors highlight the importance of scheduling Map tasks in cloud deployments of MapReduce, where input data is located on remote storage. TMaR schedules Map and Reduce tasks on servers that minimize the task finish time in each stage, respectively. The proposed dynamic partition binder for Reduce tasks in the Reduce stage reduces shuffling traffic, and TMaR + extends TMaR to improve total power consumption of the cluster, reducing it up to 12%.

Jiang et al. [79] studied MapReduce scheduling on n parallel machines with different speeds, where each job contains map tasks and reduce tasks, and the reduce tasks can only be processed after finishing all map tasks. The authors consider both offline and online scheduling problems and propose approximation algorithms with worst-case ratios for non-preemptive and preemptive reduce tasks. In the offline version, the authors propose an algorithm with a worst-case ratio of $\max\{1 + \Delta^2 - 1/n, \Delta\}$ for non-preemptive reduce tasks, where n is the number of servers, and Δ is the ratio between the maximum server speed and the minimum speed. They also design a 2-ratio algorithm for preemptive reduce tasks. In the online version, the authors devise two heuristics for non-preemptive and preemptive reduce tasks, respectively, based on the offline algorithms.

Verma et al. [80] designed a two-stage scheduler called the BalancedPools to minimize the makespan of multi-wave batch jobs. The method divides the jobs into two pools with the same makespan and assigns resources equally among the pools. Then, to minimize the makespan of each pool, the Johnson algorithm is applied within each pool to determine an order of jobs. Finally, the MapReduce simulator SimMR estimates the overall makespan for two pools.

Yao et al. [81] proposed TuMM, a slot management scheme in order to allow dynamic slot configuration in Hadoop. The scheduling goal is to utilize the resources efficiently and minimize the makespan of a batch of jobs.

In order to achieve this goal, two major modules are proposed: Workload Monitor and Slot Assigner. To predict the present workloads of map and reduce tasks, Workload Monitor periodically collects prior workload information, including execution times of completed tasks. Slot Assigner for each node utilizes the estimated information from Workload Monitor to modify the slot ratio between map and reduce. Slot ratio is utilized as a tunable knob between Map and Reduce tasks.

Zheng et al. [82] studied a joint scheduling optimization for MapReduce, which overlap the map and shuffle phases and execute simultaneously. Mitigating the average job makespan is the goal of scheduling. The main issue is that since the map and shuffle phases have a dependency relationship, they cannot be fully parallelized. Therefore, after the map phase emits data, the shuffle phase must wait for it to be transferred. To solve the above problem, the authors introduced a new concept called "strong pair." As defined by them, two jobs are considered "strong pairs" if their map and shuffle workloads are equal. They proved that when all the jobs can be broken down into strong pairs, the best schedule is to run jobs that can form a strong pair pairwise. To perform jobs in a pairwise manner, a number of offline and online scheduling policies are presented. First, jobs are classified based on their workloads. Then, using a pairwise manner, jobs are executed within each group.

Tang et al. [83] proposed an optimized scheduling algorithm for MapReduce workflow, named MRWS, in heterogeneous clusters. Workflows are modeled by DAG graphs containing MapReduce jobs. The scheduler includes a phase for prioritizing jobs and a phase for allocating tasks. First, the jobs are divided into I/O-intensive and computing-intensive categories, and each job's priorities are determined by considering its category. After that, each block is assigned a slot, and tasks are scheduled based on the data locality [95]. We described the most popular makespan-aware schedulers. Table 7 provides a summary of the main properties of makespan-aware schedulers. The analysis table includes references, key ideas, advantages and disadvantages, comparison algorithms, and evaluation techniques.

Learning-aware schedulers

In this section, we first review several important learning-aware schedulers. Then, the reviewed schedulers are compared and summarized.

Ghazali et al. [84] focused on the scheduling of MapReduce jobs in Hadoop and specifically addresses the importance of data and cache locality in improving performance. The authors propose a job scheduler called CLQLMRS (Cache Locality with Q-Learning in MapReduce Scheduler) that utilizes reinforcement learning

Table 9 Scheduling metrics in the reviewed algorithms

Reference	Meet deadline	Data locality	Makespan	Execution time	Completion time	Cost	Resource utilization	Response time	Throughput
Gao et al. [33]	•						+		
Cheng et al. [34]	•				✚				
Kao et al. [35]	•	❖							
Verma et al. [36]	•				✚				
Phan et al. [37]	•								
Kc et al. [38]	•								
Teng et al. [39]	•								
Wang et al. [40]	•						+		
Dong et al. [41]				✓					
Verma et al. [42]					✚				
Jabbari et al. [43]	•					#			
Shao et al. [44]	•			✓					
Lin et al. [45]	•								
Chen et al. [46]	•			✓					
Tang et al. [47]	•	❖			✚				
Verma et al. [48]	•				✚				
Polo et al. [49]	•	❖							
Kalia et al. [50]		❖		✓					*
Li et al. [51]		❖		✓					
Fu et al. [52]				✓					
Gandomi et al. [53]		❖			✚				
He et al. [54]		❖						■	
Ibrahim et al. [55]		❖						■	
Zhang et al. [56]		❖		✓				■	
Zhang et al. [57]		❖		✓					
Zaharia et al. [58]		❖						■	*
Seethalakshmi et al. [59]	•	❖		✓		#	+		
Tang et al. [60]	•					#			
Vinutha et al. [61]	•				✚	#			
Javanmardi et al. [62]						#			
Rashmi et al. [63]	•					#			
Zacheilas et al. [64]			○	✓		#			
Palanisamy et al. [65]						#		■	
Chen et al. [66]						#			
Aarthee et al. [67]							+		
JEYARAJ et al. [68]			○				+		
Zhang et al. [69]		❖		✓			+		
Rasooli et al. [70]		❖			✚			■	
Polo et al. [71]			○		✚		+		
Sharma et al. [72]					✚		+		
Pastorelli et al. [73]								■	
Tian et al. [74]			○						*
Ghoneem et al. [75]		❖			✚				
Varalakshmi et al. [76]			○						
Maleki et al. [77]			○	✓					

Hedayati *et al. Journal of Cloud Computing* (2023) 12:143

Page 27 of 30

Table 9 (continued)

Reference	Meet deadline	Data locality	Makespan	Execution time	Completion time	Cost	Resource utilization	Response time	Throughput
Maleki et al. [78]									
Jiang et al. [79]		○							
Verma et al. [80]		○							
Yao et al. [81]		○		✓					
Zheng et al. [82]		○		✓					
Tang et al. [83]		○							
Ghazali et al. [84]	❖			✓					
Naik et al. [85]	❖								
Naik et al. [86]					✦				
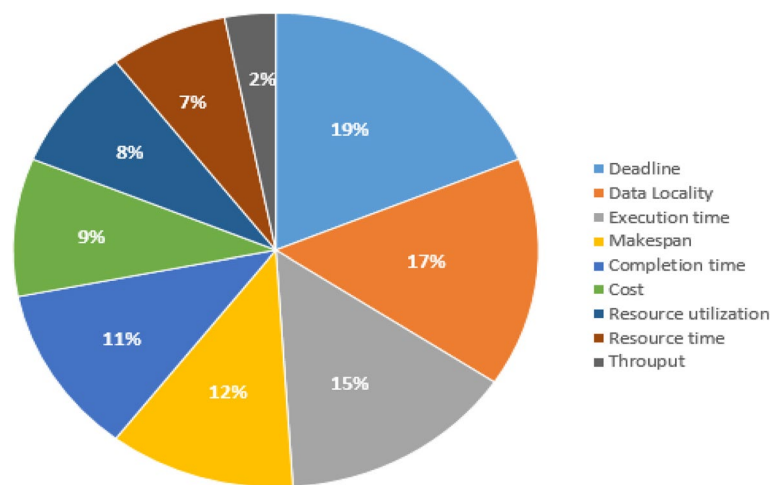
Fig. 8 Percentage of scheduling metrics in reviewed algorithms

techniques to optimize both data and cache locality. The proposed scheduler is evaluated through experiments in a heterogeneous environment. The results demonstrate a significant reduction in execution time compared to other scheduling algorithms such as FIFO, Delay, and Adaptive Cache Local. The CLQLMRS algorithm improves Hadoop performance compared to the aforementioned schedulers.

Naik et al. [85] focused on the challenges of MapReduce job scheduling in heterogeneous environments and the importance of data locality in improving the performance of the MapReduce framework. The paper highlights that data locality, which involves moving computation closer to the input data for faster access, is a critical factor in enhancing the performance of MapReduce in heterogeneous Hadoop clusters. However, the existing MapReduce framework does not fully consider heterogeneity from a data locality perspective. To address these issues, the paper proposes a novel hybrid scheduler that utilizes a reinforcement learning approach. The

proposed scheduler aims to identify true straggler tasks and schedule them on fast processing nodes in the heterogeneous cluster, taking data locality into account.

Naik et al. [86] proposed a novel MapReduce scheduler called MapReduce Reinforcement Learning (MRRL) scheduler, which leverages reinforcement learning techniques to adaptively schedule tasks in heterogeneous environments. The MRRL scheduler observes the system state of task execution and identifies slower tasks. It suggests speculative re-execution of these slower tasks on other available nodes in the cluster to achieve faster execution. The proposed approach does not require prior knowledge of the environmental characteristics and can adapt to the heterogeneous environment over time. The authors employ the SARSA learning algorithm, which is a model-free approach that solves the problem of searching optimal states with state transitions depending on the scheduler. The state determination criterion and reward function in the proposed MRRL algorithm are based on the objective of minimizing job completion time.

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 28 of 30
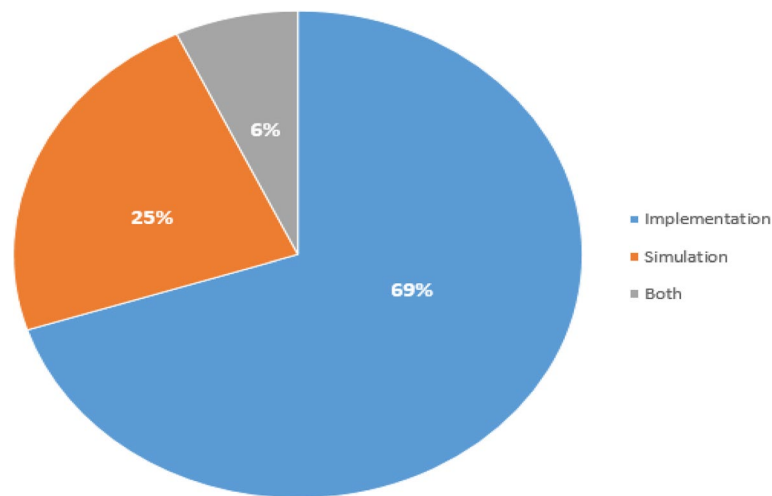
Fig. 9 Evaluation techniques used by studies addressed by the reviewed algorithms

We described several learning-aware schedulers. Table 8 provides a summary of the main properties of learning-aware schedulers. The analysis table includes references, key ideas, advantages and disadvantages, comparison algorithms, and evaluation techniques.

Discussion

To respond to RQ5, a comparative analysis of different scheduling metrics in Hadoop MapReduce is presented in this section. In Table 9 and Fig. 8, we showed how the selected studies addressed the scheduling metrics. Figure 8 demonstrates that 19% of the algorithms studied used the deadline metric, 17% used the data locality metric, 15% addressed the execution time metric, 12% addressed the makespan metric, 11% used the completion time metric, 9% used the cost metric, 8% used the response time metric, 7% used the resource utilization metric, and 2% addressed the throughput. It is shown that the majority of algorithms have focused on the deadline and data locality metrics.

Figure 9 shows the evaluation techniques used in the selected studies. As can be seen, 69% of the studies used implementation, which is the highest; 6% of them used simulation, and 25% of them used both implementation and simulation.

Conclusion

Scheduling in Hadoop MapReduce is an important challenge that Hadoop systems are facing. In this paper, we provided a comprehensive systematic study in Hadoop MapReduce. First, an overview of Hadoop major components is presented. According to our research questions, from more than 500 papers, 53 primary studies were selected. Then we thoroughly reviewed and analysed individually the selected MapReduce scheduling algorithms. Based on our research method, we classify these schedulers into six categories: deadline-aware schedulers, data locality-aware schedulers, cost-aware schedulers, resource-aware schedulers, makespan-aware schedulers, and learning-aware schedulers. We compared the studies in terms of key ideas, main objectives, advantages, disadvantages, comparison algorithms, and evaluation techniques. The results are summarized in a table in each category.

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 29 of 30

References

1. Assunção MD et al (2015) Big Data computing and clouds: Trends and future directions. J Parallel Distributed Comput 79:3–15
2. Thusoo A et al (2010) "Hive-a petabyte scale data warehouse using hadoop." 2010 IEEE 26th international conference on data engineering (ICDE 2010). IEEE
3. Deshai N et al (2019) "Big data Hadoop MapReduce job scheduling: A short survey." Information Systems Design and Intelligent Applications: Proceedings of Fifth International Conference INDIA 2018 Volume 1. Springer, Singapore
4. Hu H et al (2014) Toward scalable systems for big data analytics: A technology tutorial. IEEE Access 2:652–687
5. Chen CP, Zhang C-Y (2014) Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Inf Sci 275:314–347
6. Chen M, Mao S, Liu Y (2014) Big data: A survey. Mobile Netw Appl 19:171–209
7. Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
8. Bakni N-E and I Assayad (2021) Survey on improving the performance of MapReduce in Hadoop. In: Proceedings of the 4th International Conference on Networking, Information Systems & Security
9. Zhang B, Wang X, Zheng Z (2018) The optimization for recurring queries in big data analysis system with MapReduce. Futur Gener Comput Syst 87:549–556
10. Kashgarani H, Kotthoff L (2021) "Is algorithm selection worth it? Comparing selecting single algorithms and parallel execution." AAAI Workshop on Meta-Learning and MetaDL Challenge. PMLR
11. Pakize SR (2014) A comprehensive view of Hadoop MapReduce scheduling algorithms. Int J Comput Netw Commun Secur 2(9):308–317
12. Kang Y, Pan L, Liu S (2022) Job scheduling for big data analytical applications in clouds: A taxonomy study. Futur Gener Comput Syst 135:129–145
13. Bhosale HS, Gadekar DP (2014) Big data processing using hadoop: survey on scheduling. Int J Sci Res 3(10):272–277
14. Shvachko K et al (2010) "The hadoop distributed file system." 2010 IEEE 26th symposium on mass storage systems and technologies (MSST). Ieee
15. Khushboo K, Gupta N (2021) "Analysis of hadoop MapReduce scheduling in heterogeneous environment." Ain Shams Engineering Journal 12(1):1101–1110
16. White T (2012) Hadoop: The definitive guide. " O'Reilly Media, Inc."
17. Lu Z et al (2018) IoTDeM: An IoT big data-oriented MapReduce performance prediction extended model in multiple edge clouds. J Parallel Distributed Comput 118:316–327
18. Singh R, Kaur PJ (2016) Analyzing performance of Apache Tez and MapReduce with hadoop multinode cluster on Amazon cloud. J Big Data 3(1):1–10
19. Wang H et al (2015) BeTL: MapReduce checkpoint tactics beneath the task level. IEEE Trans Serv Comput 9(1):84–95
20. Petersen K, Vakkalanka S, Kuzniarz L (2015) Guidelines for conducting systematic mapping studies in software engineering: An update. Inf Softw Technol 64:1–18
21. Cruz-Benito J (2016) Systematic literature review & mapping
22. Lu Q et al (2015) "MapReduce job optimization: a mapping study." 2015 International Conference on Cloud Computing and Big Data (CCBD). IEEE
23. Ghazali R et al (2021) A classification of Hadoop job schedulers based on performance optimization approaches. Clust Comput 24(4):3381–3403
24. Abdallat AA, Alahmad AI, AlWidian JA (2019) Hadoop mapreduce job scheduling algorithms survey and use cases. Mod Appl Sci 13(7):1–38
25. Hashem IAT et al (2020) MapReduce scheduling algorithms: a review. J Supercomput 76:4915–4945
26. Soualhia M, Khomh F, Tahar S (2017) Task scheduling in big data platforms: a systematic literature review. J Syst Softw 134:170–189
27. Khezr SN, Navimipour NJ (2017) MapReduce and its applications, challenges, and architecture: a comprehensive review and directions for future research. J Grid Comput 15:295–321
28. Senthilkumar M, Ilango P (2016) A survey on job scheduling in big data. Cybern Inf Technol 16(3):35–51
29. Hashem IAT et al (2016) MapReduce: Review and open challenges. Scientometrics 109:389–422
30. Li R et al (2016) MapReduce parallel programming model: a state-of-the-art survey. Int J Parallel Prog 44:832–866
31. Tiwari N et al (2015) Classification framework of MapReduce scheduling algorithms. ACM Comput Surveys (CSUR) 47(3):1–38
32. Polato I et al (2014) A comprehensive view of Hadoop research—A systematic literature review. J Netw Comput Appl 46:1–25
33. Gao Y, Zhang K (2022) "Deadline-aware preemptive job scheduling in hadoop yarn clusters." 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE
34. Cheng D et al (2018) Deadline-aware MapReduce job scheduling with dynamic resource availability. IEEE Trans Parallel Distrib Syst 30(4):814–826
35. Kao Y-C, Chen Y-S (2016) Data-locality-aware mapreduce real-time scheduling framework. J Syst Softw 112:65–77
36. Verma A et al (2012) "Deadline-based workload management for MapReduce environments: Pieces of the performance puzzle." 2012 IEEE Network Operations and Management Symposium. IEEE
37. Phan LT et al (2011) "An empirical analysis of scheduling techniques for real-time cloud-based data processing." 2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA). IEEE
38. Kc K, Anyanwu K (2010) "Scheduling hadoop jobs to meet deadlines." 2010 IEEE Second International Conference on Cloud Computing Technology and Science. IEEE
39. Teng F et al (2014) A novel real-time scheduling algorithm and performance analysis of a MapReduce-based cloud. J Supercomput 69(2):739–765
40. Wang X et al (2015) SAMES: deadline-constraint scheduling in MapReduce. Front Comp Sci 9:128–141
41. Dong X, Wang Y, Liao H (2011) "Scheduling mixed real-time and non-real-time applications in mapreduce environment." 2011 IEEE 17th International Conference on Parallel and Distributed Systems. IEEE
42. Verma A, Cherkasova L, Campbell RH (2011) "Resource provisioning framework for mapreduce jobs with performance goals." Middleware 2011: ACM/IFIP/USENIX 12th International Middleware Conference, Lisbon, Portugal, December 12-16, 2011. Proceedings 12. Springer Berlin Heidelberg
43. Jabbari A et al (2021) "A Cost-Efficient Resource Provisioning and Scheduling Approach for Deadline-Sensitive MapReduce Computations in Cloud Environment." 2021 IEEE 14th International Conference on Cloud Computing (CLOUD). IEEE
44. Shao Y et al (2018) Efficient jobs scheduling approach for big data applications. Comput Ind Eng 117:249–261
45. Lin J-W, Arul JM, Lin C-Y (2019) Joint deadline-constrained and influence-aware design for allocating MapReduce jobs in cloud computing systems. Clust Comput 22:6963–6976
46. Chen C-H, Lin J-W, Kuo S-Y (2015) MapReduce scheduling for deadline-constrained jobs in heterogeneous cloud computing systems. IEEE Trans Cloud Comput 6(1):127–140
47. Tang Z et al (2013) A MapReduce task scheduling algorithm for deadline constraints. Clust Comput 16:651–662
48. Verma AL, Cherkasova, and RH Campbell (2011) Aria: automatic resource inference and allocation for mapreduce environments. In: Proceedings of the 8th ACM international conference on Autonomic computing
49. Polo J et al (2013) Deadline-based MapReduce workload management. IEEE Trans Netw Serv Manage 10(2):231–244
50. Kalia K et al (2022) Improving MapReduce heterogeneous performance using KNN fair share scheduling. Robot Auton Syst 157:104228
51. Li Y, Hei X (2022) "Performance optimization of computing task scheduling based on the Hadoop big data platform." Neural Computing and Applications pp. 1-12
52. Fu Z et al (2020) An optimal locality-aware task scheduling algorithm based on bipartite graph modelling for spark applications. IEEE Trans Parallel Distrib Syst 31(10):2406–2420
53. Gandomi A et al (2019) HybSMRP: a hybrid scheduling algorithm in Hadoop MapReduce framework. J Big Data 6:1–16
54. He C, Lu Y, Swanson D (2011) "Matchmaking: A new mapreduce scheduling technique." 2011 IEEE Third International Conference on Cloud Computing Technology and Science. IEEE

Hedayati *et al. Journal of Cloud Computing*   (2023) 12:143

Page 30 of 30

55. Ibrahim S et al (2012) "Maestro: Replica-aware map scheduling for mapreduce." 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012). IEEE

56. Zhang X et al (2011) "An effective data locality aware task scheduling method for MapReduce framework in heterogeneous environments." 2011 International Conference on Cloud and Service Computing. IEEE

57. Zhang X et al (2011) "Improving data locality of mapreduce by scheduling in homogeneous computing environments." 2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications. IEEE

58. Zaharia M et al (2010) Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In: Proceedings of the 5th European conference on Computer systems

59. Tang X et al (2021) Cost-efficient workflow scheduling algorithm for applications with deadline constraint on heterogeneous clouds. IEEE Trans Parallel Distrib Syst 33(9):2079–2092

60. Seethalakshmi V, Govindasamy V, Akila V (2022) Real-coded multi-objective genetic algorithm with effective queuing model for efficient job scheduling in heterogeneous Hadoop environment. J King Saud Univ-Computer Inf Sci 34(6):3178–3190

61. Vinutha D, Raju G (2021) Budget constraint scheduler for big data using Hadoop MapReduce. SN Comput Sci 2(4):250

62. Javanmardi AK et al (2021) A unit-based, cost-efficient scheduler for heterogeneous Hadoop systems. J Supercomput 77:1–22

63. Rashmi S, Basu A (2016) "Deadline constrained Cost Effective Workflow scheduler for Hadoop clusters in cloud datacenter." 2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS). IEEE

64. Zacheilas N, Kalogeraki V (2016) "Chess: Cost-effective scheduling across multiple heterogeneous mapreduce clusters." 2016 IEEE international conference on autonomic computing (ICAC). IEEE, Berahmand, [10/4/2023 8:36 PM]

65. Palanisamy B, Singh A, Liu L (2014) Cost-effective resource provisioning for mapreduce in a cloud. IEEE Trans Parallel Distrib Syst 26(5):1265–1279

66. Chen K et al (2013) CRESP: Towards optimal resource provisioning for MapReduce computing in public clouds. IEEE Trans Parallel Distrib Syst 25(6):1403–1412

67. Aarthee S, Prabakaran R (2023) Energy-aware heuristic scheduling using bin packing mapreduce scheduler for heterogeneous workloads performance in big data. Arab J Sci Eng 48(2):1891–1905

68. Jeyaraj R, Paul A (2022) Optimizing MapReduce task scheduling on virtualized heterogeneous environments using ant colony optimization. IEEE Access 10:55842–55855

69. Zhang Q et al (2015) PRISM: Fine-grained resource-aware scheduling for MapReduce. IEEE Trans Cloud Comput 3(2):182–194

70. Rasooli A, Down DG (2014) COSHH: A classification and optimization based scheduler for heterogeneous Hadoop systems. Futur Gener Comput Syst 36:1–15

71. Polo J et al (2011) "Resource-aware adaptive scheduling for mapreduce clusters." Middleware 2011: ACM/IFIP/USENIX 12th International Middleware Conference, Lisbon, Portugal, December 12-16, 2011. Proceedings 12. Springer, Berlin Heidelberg

72. Sharma B et al (2012) "Mrorchestrator: A fine-grained resource orchestration framework for mapreduce clusters." 2012 IEEE Fifth International Conference on Cloud Computing. IEEE

73. Pastorelli M et al (2015) HFSP: bringing size-based scheduling to hadoop. IEEE Trans Cloud Comput 5(1):43–56

74. Tian F, Chen K (2011) "Towards optimal resource provisioning for running mapreduce programs in public clouds." 2011 IEEE 4th International Conference on Cloud Computing. IEEE

75. Ghoneem M, Kulkarni L (2017) "An adaptive MapReduce scheduler for scalable heterogeneous systems." Proceedings of the International Conference on Data Engineering and Communication Technology: ICDECT 2016, Volume 2. Springer Singapore, Berahmand, [10/4/2023 8:40 PM]

76. Varalakshmi P, Subbiah S (2022) Optimized scheduling of multi-user MapReduce jobs in heterogeneous environment. Concurr Comput: Pract Exp 34(27):e7316

77. Maleki N, Rahmani AM, Conti M (2021) SPO: a secure and performance-aware optimization for MapReduce scheduling. J Netw Comput Appl 176:102944

78. Maleki N et al (2020) TMaR: a two-stage MapReduce scheduler for heterogeneous environments. HCIS 10:1–26

79. Jiang Y et al (2017) Makespan minimization for MapReduce systems with different servers. Futur Gener Comput Syst 67:13–21

80. Verma A, Cherkasova L, Campbell RH (2013) Orchestrating an ensemble of MapReduce jobs for minimizing their makespan. IEEE Trans Dependable Secure Comput 10(5):314–327

81. Yao Y et al (2015) Self-adjusting slot configurations for homogeneous and heterogeneous hadoop clusters. IEEE Trans Cloud Comput 5(2):344–357

82. Zheng H, Wan Z, Wu J (2016) "Optimizing MapReduce framework through joint scheduling of overlapping phases." 2016 25th International Conference on Computer Communication and Networks (ICCCN). IEEE

83. Tang Z et al (2016) An optimized MapReduce workflow scheduling algorithm for heterogeneous computing. J Supercomput 72:2059–2079

84. Ghazali R et al (2022) CLQLMRS: improving cache locality in MapReduce job scheduling using Q-learning. J Cloud Comput 11(1):1–17

85. Naik NS, Negi A (2017) "A learning-based mapreduce scheduler in heterogeneous environments." 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE

86. Naik NS, Negi A, Sastry V (2015) Performance improvement of MapReduce framework in heterogeneous context using reinforcement learning. Proc Comput Sci 50:169–175

87. Varga M, Petrescu-Nita A, Pop F (2018) Deadline scheduling algorithm for sustainable computing in Hadoop environment. Comput Secur 76:354–366

88. He C, Lu Y, Swanson D (2013) Real-time scheduling in mapreduce clusters. In: 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing. IEEE

89. Gautam JV et al (2015) "A survey on job scheduling algorithms in big data processing." 2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT). IEEE

90. Chen CH, Lin JW, Kuo SY (2014) "Deadline-constrained MapReduce scheduling based on graph modelling." 2014 IEEE 7th International Conference on Cloud Computing. IEEE

91. Nimbalkar PP, Gadekar DP (2015) Survey on scheduling algorithm in mapreduce framework. IJSETR 4(4):1226–1230

92. Singh N, Agrawal S (2015) A review of research on MapReduce scheduling algorithms in Hadoop." International Conference on Computing, Communication & Automation. IEEE

93. Khan M et al (2015) Hadoop performance modeling for job estimation and resource provisioning. IEEE Trans Parallel Distrib Syst 27(2):441–454

94. Mohamed E, Hong Z (2016) "Hadoop-MapReduce job scheduling algorithms survey." 2016 7th International Conference on Cloud Computing and Big Data (CCBD). IEEE

95. Mittal R and H Kaur A Survey on Data Placement and Workload Scheduling Algorithms in Heterogeneous Network for Hadoop. Int J Comput Appl 975:8887

Publisher's Note