Arabian Journal for Science and Engineering
https://doi.org/10.1007/s13369-024-09779-9

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

Scheduling of Big Data Workflows in the Hadoop Framework with Heterogeneous Computing Cluster

Amir Masoud Rahmani[1] · Ehsan Yazdani Chamzini[2,3] · Mohsen pourshaban[2,3] · Mehdi Hosseinzadeh[4,5]

Received: 2 May 2024 / Accepted: 12 November 2024

Abstract
Recently, resource allocation in cloud computing has become a popular research topic. Hi-WAY is a scientific workflow management system that facilitates workflows involving large-scale inputs such as big data. Hadoop, a framework designed to implement distributed systems, allows Hi-WAY to be run on thousands of computing nodes with desirable fault tolerance. Task scheduling is not difficult in a homogeneous Hadoop system, where computing nodes have identical specifications. However, task scheduling could be problematic in heterogeneous systems, where specifications such as processor power, memory, and bandwidth may vary from node to node. This paper introduces a workflow scheduler on the Hadoop framework (WSH), accounting for system heterogeneity when scheduling computing- and IO-intensive jobs. WSH uses a training task to collect information before distributing jobs. The results demonstrate effective job allocation and load balancing improvement in Hadoop, leading to increased resource efficiency and reduced makespan. Based on various experiments and the use of different workflows, the proposed method improves the scheduling length ratio by 42%, reduces makespan by 20%, and enhances speedup by approximately 37% compared to the algorithm.

Keywords Hadoop · Hi-WAY · Big data · Workflow · Scheduling

1 Introduction

The massive volume, variety, and rapid evolution of big data generated by IoT devices, social networks, and scientific organizations present challenges that traditional computational processes cannot handle effectively [28]. These challenges include ensuring system scalability to manage vast data volumes, handling network latency, managing uneven workload distribution across computational resources, and precisely managing dependencies between processing stages. Additionally, resource heterogeneity and data localization are essential for optimizing system performance. Furthermore, fault tolerance and unpredictable data volume fluctuations demand flexible scheduling approaches. Alongside the need for energy optimization and real-time processing, these factors collectively make big data workflow scheduling a complex and crucial issue in distributed systems. To address these challenges, programming models like MapReduce and frameworks such as Hadoop have been developed, significantly enhancing parallel processing capabilities in cloud and distributed environments and overcoming the limitations of traditional approaches [15, 23, 26].

Hadoop is an open-source software framework written in Java, designed to allow parallel processing on thousands of machines with high fault tolerance [6]. Many large corporations, including Facebook and Yahoo, are currently using

✉ Mehdi Hosseinzadeh
mehdihosseinzadeh@duytan.edu.vn

Amir Masoud Rahmani
rahmania@yuntech.edu.tw

Ehsan Yazdani Chamzini
Ehsan.yazdani@sco.iaun.ac.ir

Mohsen pourshaban
Pourshaban@sco.iaun.ac.ir

1 Future Technology Research Center, National Yunlin University of Science and Technology, Yunlin, Taiwan

2 Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran

3 Big Data Research Center, Najafabad Branch, Islamic Azad University, Najafabad, Iran

4 School of Computer Science, Duy Tan University, Da Nang, Vietnam

5 Jadara University Research Center, Jadara University, Irbid, Jordan

Published online: 25 November 2024

Hadoop. Nowadays, social networks and scientific organizations generate petabytes of data every week [9, 17]. Researchers must create pipelines and incorporate numerous tools to link tasks to manage these huge amounts of data. Scientific workflow management systems have been developed to facilitate the processing of these data. Hi-WAY is a scientific workflow management system implementing a black-box view of tools, tasks, and workflow data. Thus, it allows every task in the workflow graph, from a simple script to various external services, to be processed in Hadoop [4, 5].

Directed acyclic graph (DAG) scheduling for the proper task allocation to heterogeneous computing nodes in distributed systems is an NP-Hard problem [12]. In [34], a heuristic method is proposed to schedule Heterogeneous Earliest Finish Time (HEFT) in multiprocessor systems, where tasks are explicitly prioritized in the workflow graph based on upward ranking. Then, in the processor allocation phase, a task is allocated to a processor such that the predefined cost, i.e., makespan, is minimized. In [33], an optimum scheduling policy, called MapReduce-enabled workflow scheduler (MRWS), is presented, which incorporates the idea of the HEFT method [34]. In this method, job scheduling in a workflow is accomplished by turning jobs into tasks such that these tasks can be distributed over proper slots. Jobs that require more processing are called computing-intensive, and other jobs are called IO-intensive. In order to achieve better performance, several tasks of a job could be scheduled in slots' idle time, even if they are shorter than the time required to complete the job. This approach results in higher efficiency and shorter runtimes. This paper presents a method for processing workflow graphs with big data, where the tasks in the workflow graph are allocated based on the computing nodes' running ability.

When implementing Hadoop in heterogeneous environments, poor data blocks and task distribution may lead to overload or underload and higher response time on some computing nodes. This is because faster computing nodes process their data blocks and call them from slower nodes, causing heavy traffic on the network bandwidth and multiple runs of speculative tasks. A few algorithms are proposed for scheduling big data workflows in heterogeneous environments. Because of the scarcity of scheduling policies in the Hadoop framework, the algorithms in this field should be able to consider parallel processing of jobs in the workflow graph, distribution of tasks based on types of jobs (computing-intensive and IO-intensive), data locality, distribution of data blocks based on computing powers of nodes, storage capacities, and network structure.

Contributions of this study are as follows:

- Our proposed workflow scheduler (WSH) handles diverse resource requirements in Hadoop environments and focuses on addressing the challenge of task scheduling in heterogeneous Hadoop systems.
- By using training tasks to gather system information before job distribution, our workflow scheduler enhances its adaptability to dynamic environments.

The rest of this paper is structured as follows: In Sect. 2, the methods and algorithms on MapReduce and workflows in the heterogeneous computing cluster will be reviewed. In Sect. 3, the proposed method will be explained in detail. In Sect. 4, the implementation results will be compared and analyzed, and Sect. 5 concludes the paper.

2 Related Work

Scheduling is a technique for allocating resources to jobs to minimize starvation and maximize resource utilization. One way to improve scheduling performance is to set deadlines for jobs. Scheduling algorithms for MapReduce jobs are divided into two general groups, inbuilt and user-defined, consisting of several methods. Three default inbuilt schedulers are available in the Hadoop framework: first in first out (FIFO), capacity, and fair [35]. Hi-WAY consists of inbuilt static and dynamic schedulers. Dynamic schedulers include methods such as first come first served (FCFS), and two examples of static schedulers are round-Robin and HEFT [4]. The user-defined scheduling algorithms for MapReduce jobs and others that may improve performance in this context are reviewed in [27].

In [8], Rahmani et al. introduce an ant colony optimization algorithm for cloud computing applications, where scheduling overheads in dynamic environments are reduced. In [13], several case studies are performed on a select group of job scheduling and load balancing algorithms developed for clouds, and these algorithms are classified into multiple subgroups.

In the algorithm presented in [11], virtual machines are classified into virtual clusters, and load balancing and workload are improved by taking replications into account. The authors [2] presented a dynamic scheduling algorithm incorporating the hill-climbing algorithm for load balancing and makespan reduction. The conflict between specifications of physical devices and user requests leads to failure to guarantee service quality in the pass layer for customers and lower energy efficiency for cloud service providers. In [3], a particle swarm optimization-based method is proposed to reallocate migrated virtual machines in the overloaded host to tackle this issue. In [37] HEMS algorithm is proposed in hybrid cloud environment that consists of five components: workflow scheduling sequence generation, task scheduling sequence initialization for each workflow, optimal scheduling scheme determination for each task, initial task scheduling sequence

optimization, and optimal scheduling plan optimization for minimizing the total electricity cost of all servers. In [25] HGSOA-GOA algorithm is proposed for scientific workflow scheduling problems in multi-cloud computing environments for optimizing factors such as makespan, cost, energy, and throughput. In [24] Mikram et al. introduce the Hybrid HEFTPSO-GA algorithm (HEPGA), aiming to efficiently allocate tasks to available resources across the datacenter and minimizing the makespan. In [24] a hybrid scheduling algorithm (PCP–ACO) is introduced, consists of two phases: task ordering and resource selection. This aims to minimize the execution cost of a workflow in cloud environments. In [38] Xue et al. introduce a deep reinforcement learning-based workflow scheduling in Hadoop that can significantly reduce the average job latency.

Table 1 presents a shortlist of workflow scheduling algorithms available for the cloud and their workflow type and running environment.

The purpose of this study is to consider computing clusters in a heterogeneous manner, as these clusters comprise processing, storage, and network resources with varying capacities and accessibility. Consequently, we aim to cluster heterogeneous processing resources, given that some tasks are computationally intensive while others are I/O intensive. By providing the scheduler with information about the effectiveness of heterogeneous resources, we can assign tasks to appropriate compartments, enabling each node to deliver high performance.

3 The Proposed Workflow Scheduler

In this research, the processing of big data workflows is considered a DAG consisting of multiple jobs. In a heterogeneous cloud computing setting, computation speed may vary from node to node and even from job to job in one node, which is likely to cause a load imbalance. Hence, we propose an algorithm for big data workflow scheduling in Hadoop, which considers the nodes' heterogeneity when scheduling the Hi-WAY jobs and controlling their running process. This algorithm consists of two stages: job prioritization and job allocation. First, all jobs' priorities are computed, and then, the proper cluster for the job allocation is determined by running a training task on the first node of every cluster. Our scheduling algorithm incorporates some of the MRWS algorithm's policies [33] and the HEFT algorithm into Hi-WAY. In MRWS, the big data workflow consists of several MapReduce jobs.

When possible, each job is divided into tasks. Otherwise, the whole job is considered a task. In WSH, each job in the workflow graph consists of many tasks, and each task is regarded as a Hi-WAY job. WSH schedules the job by running the training task on the first node of each cluster. In this paper, the jobs requiring resources with high processing power are considered computing-intensive, and other jobs are considered IO-intensive. We try to minimize makespan and maximize resource efficiency with the insertion's help of the idle time policy [34] by running some jobs between others.

3.1 The Proposed Workflow Model

The proposed workflow graph is represented by WF and consists of the following components:

$$WF = (job,\ edge,\ type)$$

where a job represents the jobs in the workflow graph, an edge represents the set of edges between the jobs in Hi-WAY (each edge has a value representing the communication cost, which in the proposed method is ignored). The type describes the type of each Hi-WAY job (computing-intensive/IO-intensive) once run, which helps us allocate jobs to nodes. Suppose some clusters have equal finish times after running the training task on the first node of clusters. In that case, the job will be classified as IO-intensive and allocated to the cluster with the lowest processing power so that more powerful nodes remain available for computing-intensive jobs.

3.2 Clustering of Heterogeneous Resources in the Proposed Method (WSH)

The jobs need processing and IO resources. However, when the heterogeneous computational clusters, computing power differences lead to unbalanced loads and prolonged response times. In WSH, a heterogeneous cluster is divided into subclusters of nodes such that each subcluster has equal processing power and memory. Figure 1 shows the proposed procedure for clustering heterogeneous resources and the workflow scheduler model.

Step 1: Cluster dividing into subclusters—In WSH, a heterogeneous cluster is divided into subclusters of nodes, each subcluster having uniform processing power and memory capacity. This heterogeneous cluster may include resources from a private cloud or leased resources from a public cloud; however, for the purposes of this study, private cloud resources are assumed.

Step 2: Job Prioritization—Jobs within Hi-WAY are prioritized based on upward ranking, ensuring optimized processing and resource allocation.

Step 3: Training Task— The scheduler executes a training task to collect data on the performance and effectiveness of the heterogeneous resources, enabling optimized allocation of jobs to appropriate nodes.

Table 1 Comparison of workflow scheduling algorithms

Implementation/simulation environment	Environment	Type of workflow	Author(s)
CloudSim	Cloud	Scientific workflows	[22], Mikram et al., 2024
WorkflowSim	hybrid cloud	Scientific workflows	[20], Shuang Wang et al., 2024
GridSim	Grid	Simple workflows	[24], Moheb et al., 2024
WorkflowSim	Cloud	Scientific workflows	[25],Yang et al., 2024
Python	Cloud	Simple workflows	[26], Choudhary., 2024
WorkflowSim CloudSim	hybrid cloud	Scientific workflows	[21], Mohammadzadeh et al., 2023
Python	cloud	Scientific workflows	[22], Shobeiri et al., 2023
Nextflow, Kubernetes	Cloud	Scientific workflows	[27], Lehmann et al., 2023
Hadoop	Cloud	Scientific workflows	[23], Xue et al., 2023
CloudSim	Cloud-Fog	Scientific workflows	[28], Rizvi et al., 2023
Hadoop	Cloud	Scientific workflows	[29], Koo et al., 2022
Nextflow, Kubernetes	Cloud	Scientific workflows	[30], Bader et al., 2021
Amazon EC2, Google cloud engine, Microsoft Azure	Cloud	Scientific workflows	[18], Mutaz et al., 2019
Hadoop, Spark	Cloud	Scientific workflows	[6], Kouanou et al., 2018
Hadoop	Cloud	Simple workflows	[31], Liu et al., 2018
Hadoop, Hi-WAY	Cloud	Scientific workflows	[8], Marc et al., 2017
CloudSim	Cloud	Scientific workflows	[32], Li et al., 2016
Hadoop	Cloud	Scientific workflows	[33], Bikas et al., 2015
Hadoop	Cloud	Scientific workflows	[34], Krish et al., 2014
Hadoop	Cloud	Scientific workflows	[12], Zhou et al., 2014
Hadoop	Cloud	Scientific workflows	[35], Li et al., 2014
Hadoop	Cloud	Scientific workflows	[36], Roshan et al., 2013

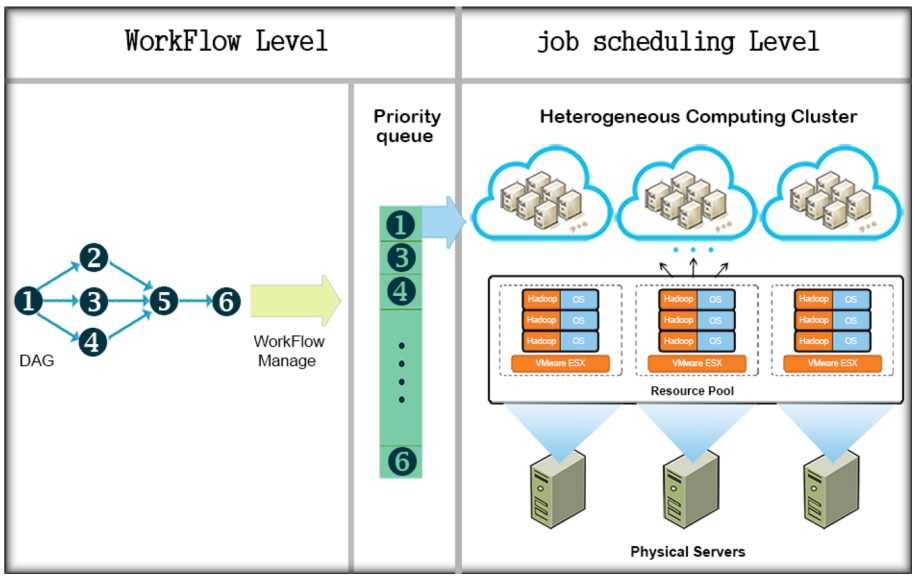

Fig. 1 A heterogeneous computational cluster and the proposed workflow scheduling model

Table 2 Codes on different levels

Programming framework	Layered structure	
Job distribution level	DAG analysis	Java
	Resource allocation	
Hi-WAY level	Scheduling of the training tasks and jobs	Java
Cluster level	Clustering of heterogeneous resources	Java

Step 4: Actual Scheduling of Hi-WAY Jobs—Finally, the actual scheduling of Hi-WAY jobs is conducted based on the data obtained from the training task. This information assists the scheduler in making optimal decisions regarding job allocation to suitable nodes.

3.3 The programming Language of the Proposed Algorithm

Table 2 shows the programming language for scheduling and distributing jobs, DAG analysis, and cluster creation.

3.4 Workflows

In this study, three workflow graphs illustrate the proposed method's work: Gene2life, Avianflu_small, and Epigenomics, with 8, 104, and 100 Hi-WAY jobs. Gene2life is a workflow typically used to analyze molecular biology [29]. This workflow receives a DNA sequence and searches for its match in the database and correlations based on function and organism. Figure 2 shows the Hi-WAY jobs of the Gene2life workflow graph. The workflow Avianflu_small is derived from the Avian-Flu Grid project, which develops a global infrastructure for studying Avian-Flu as an infectious agent and a pandemic threat. Figure 3 shows the workflow used in this project's drug design process to understand host selectivity and drug resistance mechanisms. The workflow has several small preprocessing steps followed by a final step where up to 1,000 parallel tasks are spawned. The data products of this workflow are small. The third workflow is related to the Epigenome center, which maps human cells' epigenetic state on a genome-wide scale. Epigenomic workflow is a parallel data processing pipeline whereby the Pegasus workflow management system is utilized to automate a series of gene sequence operations [14]. The tasks of this workflow are illustrated in Fig. 4.

3.5 Job Prioritization in the Workflow Graph

In WSH, DAG jobs are prioritized using Eqs. 1, 2, and 3 [33]. It should be noted that the communication cost is ignored

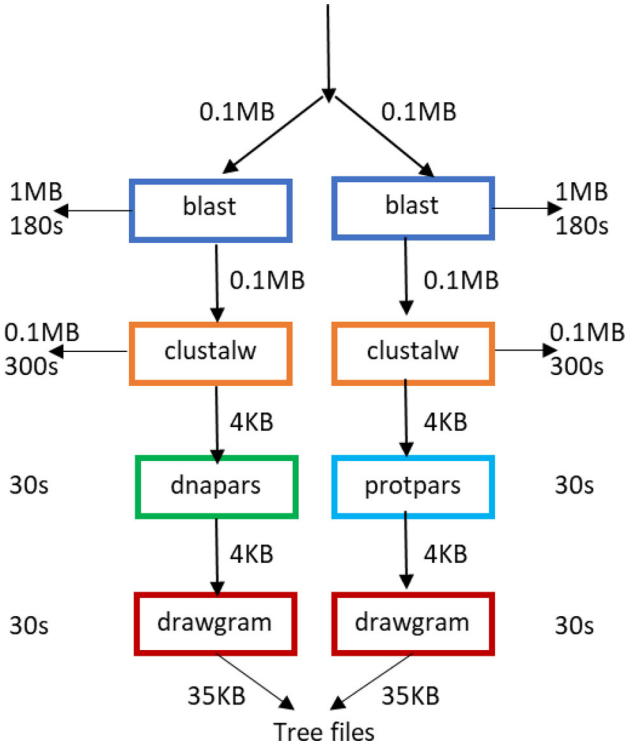

Fig. 2 Tasks of Gene2life workflow graph [29]

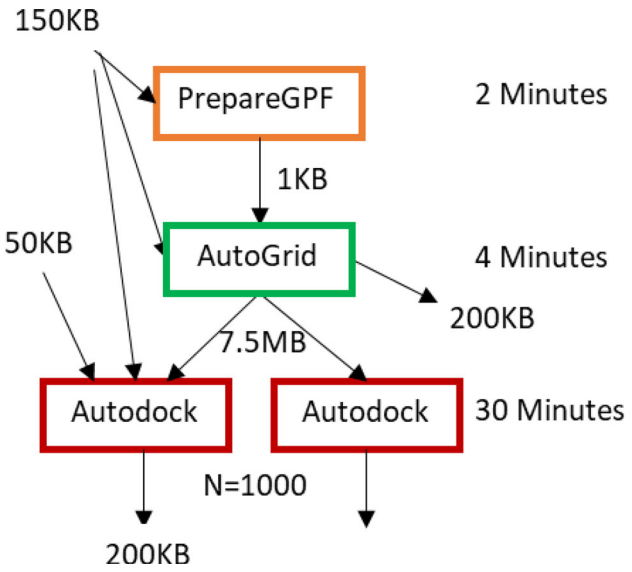

Fig. 3 Tasks of Avianflu_small workflow graph [29]

for simplicity in the proposed method, WSH. The upward ranking is defined as Eqs. 1 and 2.

$$\text{Upward}_{\text{Rank}}\left(\text{job}_{(i)}\right) = \overline{W_i} + \max_{\text{job}_{(j)} \in \text{Successor}(\text{job}_{(i)})} \left(C_{(i,j)} + \text{Upward}_{\text{Rank}}\left(\text{job}_{(j)}\right)\right) \quad (1)$$

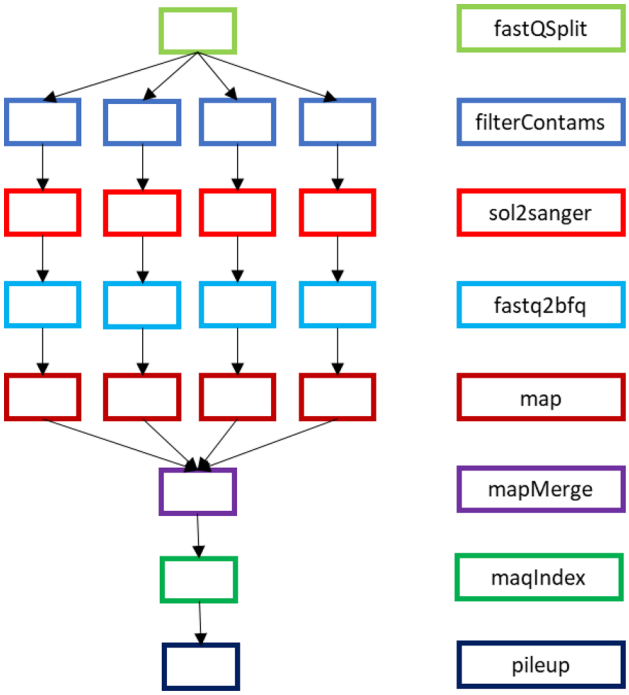

Fig. 4 Tasks of Epigenomics workflow graph [29]

Table 3 Calculation of upward rank

HI-WAY JOB	$\text{UPWARD}_{\text{RANK}}$
Blast1	1
Blast2	2
Clustalw	3
Clustalw	4
Dnapars	5
Protpars	6
Drawgram1	7
Drawgram2	8

$$\text{Upward}_{\text{Rank}}(\text{job}_{\text{exit}}) = \overline{W_{\text{exit}}} \tag{2}$$

In these equations, $\overline{W_{\text{I}}}$ is the average of computation costs of $\text{job}_{(\text{i})}$, $C_{(\text{i,j})}$ is the *communication cost* from $\text{job}_{(\text{i})}$ to $\text{job}_{(\text{j})}$, which is zero, and Successor$(\text{job}_{(i)})$ is the set of jobs that come immediately after $\text{job}_{(\text{i})}$. The exit Hi-WAY job is a job without a child. The upward ranking is calculated by traversing the graph from the exit Hi-WAY job. Table 3 shows the *upward ranking* of each Hi-WAY job in Gene2life.

The jobs in each subcluster have different finish times. The average computation cost can be obtained by running the job on the first node in each cluster and then averaging over the clusters. Table 4 shows each Hi-WAY job's computational cost in the Gene2life workflow graph on a computing node.

Table 4 Computational costs of each job on a computing node

Type of computing node	Hi-WAY job	Computational cost
Virtual machine	Blast1	01:02
	Blast2	01:02
	Clustalw	01:30
	Clustalw	01:30
	Dnapars	00:19
	Protpars	00:16
	Drawgram1	00:18
	Drawgram2	00:18

(The time is in minutes and seconds.) This table will be used for sorting the clusters and calculating the upward rank of Hi-WAY jobs.

3.6 The Resource Allocation Policy of the Proposed Method

For each cluster, a variable is defined to show the number of nodes that have executed a job. In order to find the node that would minimize the finish time, the scheduler first assigns the training task to every cluster's first nodes to run. The NodeManager reports the finish times to the ResourceManager. The scheduler then sorts the clusters based on these times. In order to run the job, the variable of each cluster is assessed, and then a set is defined. The nodes of clusters are added to this collection based on the policies outlined in Fig. 5 to select the node with the quickest finishing time. The variables used in running the training task, prioritizing the clusters, and selecting the best node are:

$\text{FT}[\text{Task}_{(i)}, c_t]$*:* the finish time of the training task on the container c_t.

$\text{EST}[\text{job}_{(i)}, c_t]$ and $\text{EFT}[\text{job}_{(i)}, c_t]$*:* the earliest start time and the earliest finish time of job_{i}.

$\text{AST}_{\text{job}(i)}$ and $\text{AFT}_{\text{job}(i)}$: the actual start time and the actual finish time of $\text{job}_{(\text{i})}$.

For the node that reduces the finish time of the Hi-WAY job more than any other considered node, $\text{AST}_{\text{job(i)}}$ and $\text{AFT}_{\text{job(i)}}$ will be set to values of $\text{EST}_{\text{job(i)}}$ and $\text{EFT}_{\text{job(i)}}$, respectively.

For the first input Hi-WAY job in DAG,

$\text{AST}_{\text{job(1)}} = \text{EST}_{\text{job(1)}}$=0.

The values of $\text{EST}_{\text{job}(i)}$*and* $\text{EFT}_{\text{job}(i)}$ are calculated from the recursive relations 3, 4, and 5, derived from [33].

$$\text{EST}[\text{job}_{(i)}, \text{c}_t] = \max\{\text{readyTime}(\text{job}_{(i)}), \text{containeravailTime}(c_t)\} \tag{3}$$

$$\text{EFT}[\text{job}_{(i)}, \text{c}_t] = \text{EST}_{\text{Job}(i)} + w(\text{job}_{(i)}, c_t) \tag{4}$$

Fig. 5 The proposed scheduling algorithm, WSH

1: ***procedure*** *Upward$_{Rank}$_Order$_{Rank}$ (WF, Some information Like Table 4)*
2: *Compute$_{Rank}$();*
3: *Sort$_{Rank}$();*
4: ***end Procedure***

$$\text{readyTime}\left(\text{job}_{(i)}\right) = \max_{\text{job}_{(j)} \in \text{predecessor}(\text{job}_{(i)})} \left(\text{AFT}_{\text{job}(j)}\right) \tag{5}$$

In order to calculate $\text{EST}_{\text{job}(i)}$, every job immediately before $\text{job}_{(i)}$ should be run. It should also be determined in which time interval the container is available to run jobs. The predecessor($\text{job}_{(i)}$) is the set of jobs immediately before job_i. readyTime(job_i) represents the time at which all of the data needed for $\text{job}_{(i)}$ are available and equals the maximum finish times of all jobs immediately before job_i. containerAvailTime(c_t) is when the container c_t is finished with the previous job, and is ready to run $\text{job}_{(i)}$. After the scheduling of $\text{job}_{(i)}$ on the container that minimizes the finish time, and thus obtaining the actual finish time EFT[$\text{job}_{(i)}$, c_m] the actual start time,$\text{AFT}_{\text{job}(i)}$:, can be obtained by the following equation:

$$\text{AST}_{\text{job}i} = \text{AFT}_{\text{job}i} - w(\text{job}_i, c_t)$$

Figure 6 shows the process of the proposed scheduler, WSH, which consists of three subprocesses.

The first subprocess is the algorithm shown in Fig. 7. In this algorithm, the upward ranks of all Hi-WAY jobs are first calculated, and then sorted in ascending order using the $\text{Order}_{\text{Rank}}()$ function.

Figure 8 displays the algorithm used to run the training tasks and sort the clusters for Hi-WAY jobs. Here, WF is the workflow, job_priority_queue is the priority queue of the Hi-WAY jobs, Clus_set() is the list of available clusters, job_Clus_sort is the array in which the sorted clusters for each job are stored based on the finish time of the training task, and Task_list() is the list of the tasks of a job in the workflow graph. In steps 8 and 9, the job is selected based on its priority from the top of the job queues. In step 10, each job's first task from the task list is determined. In steps 11 and 12, the task is assigned to the first node of each cluster, and in steps 13 and 14, the obtained finish times are added to the array jobs_Clus_sort. In step 16, the jobs_Clus_sort array is sorted for the considered job. Finally, in step 17, the Hi-WAY job is deleted from the priority list of jobs.

Figure 5 shows the algorithm of Hi-WAY job scheduling based on cluster priorities. Jobs_Clus_sort is the list of clusters sorted for jobs, and the list of nodes is the list of nodes that should be considered for selecting the best node that minimizes the finish time of the job. Node_A is the list of nodes that have run a job. Node_A is the node that has not yet run a job. Step 10 determines whether there is a job that should be scheduled.

In step 11, the job with the highest priority is selected to be run. In step 12, the list of prioritized clusters based on the training task is extracted, and the cluster with the highest priority is chosen. In steps 13 to 16, if the cluster nodes have executed a job, they are added sequentially to list of nodes. In steps 17 to 21, if nodes in the cluster have not completed a job, one such node is added to the list of nodes alongside nodes that have executed a job. All nodes in list of nodes are evaluated in steps 23 to 25 to determine the optimal node for minimizing the job's finish time. In steps 26 and 27, the job is assigned to the node with the best finish time, and the earliest finish time is stored as the actual finish time. In steps 28 and 29, if the node that minimizes the finish time of the job is the last in list of nodes, it is added to the set of nodes that have run a job. In steps 30 to 33, all nodes will be declared available if all nodes are already selected based on their optimality, and the final cluster (the one with the lowest priority) reaches the next job. The comparison for selecting the best node will start from the clusters with the highest priority. Finally, in step 34, list of nodes will be cleared, and in step 35, the scheduled job will be deleted from the job priority queues.

4 Case Study: *Optimization of Avian-Flu Task Scheduling in the Data Center*

The Avian-Flu project focuses on analyzing data related to the outbreak of avian influenza. Given the large volume and complexity of the data, optimizing task scheduling for efficient processing of this information is crucial.

4.1 Challenges

Volume and Complexity of Data: The data related to the avian influenza outbreak are continuously increasing and encompasses various types of information.

Heterogeneity of Resources: The data center utilizes a heterogeneous cluster composed of nodes with different specifications. This complicates scheduling and task allocation.

4.2 Solutions

Cluster dividing: The heterogeneous cluster was divided into subclusters, each with similar processing power and

Fig. 6 The upward ranking calculation algorithm

```
1: procedure WSH (WF, Set of predict parameters)
2: Definition Arguments
3:                WF→ WorkFlow
4: Upward_Rank_Order_Rank();
5: Training_Task_Order_cluster();
7: job Scheduling();
8: end Procedure
```

Fig. 7 The algorithm for sorting clusters and running the training task

```
1: procedure Training_Task_Order_cluster(WF, job_priority_queue, Clus_set)
2: Definition of Argumants{
3:     job_priority_queue→job priority queue for prioritizing jobs
4:     FT[Task_(i),node] → Finish Time of Training Task on node
5:     Task_list→ List of Tasks
6:     job_Clus_sort[clus_(i), task] → an Array keeps Finish Time of Training
                Task in the clus_i
7:     Clus_set→an Array keeps all the Clusters names }
8:  while there are unscheduled jobs in the job_priority_queue do
9:   Select the first job_j from job_priority_queue to prioritize Clusters:
10:      Select the first Task in Task_List() of the selected job;
11:      for each clus_i in the Clus_set() do
12:        Select the first node from clus_i;
13:        Compute FT[Task_(i), node];
14:        Add to the job_j_ Clus_sort[clus_(i),FT_TASK];
15:      end for
16:      Sort job_j_Clus_sort by FT_TASK;
17:     Remove the selected job from the job_priority_queue;
18:  end while
19: end procedure
```

memory capacity. This dividing improved resource allocation and reduced latency.

Task Prioritization: Tasks related to the Avian-Flu workflow were prioritized based on an upward ranking method. This action helped determine the precedence of each task.

Execution of a Training Task: A training task was executed to gather information about the performance and effectiveness of the resources. This information was utilized to optimize task allocation to suitable nodes.

Actual Scheduling: The actual scheduling of Avian-Flu tasks was conducted based on the data obtained from the training task, leading to increased efficiency and reduced overall processing time.

4.3 Results

By implementing these steps, task scheduling was significantly optimized, leading to increased efficiency, reduced workflow execution time, and enhanced resource effectiveness.

5 Implementation and Analysis

Some traditional workflow management systems, including Pegasus, Taverna, Galaxy, Nextflow, and Toil, can use Hi-WAY as a complementary system. This is also true for distributed workflow management systems such as Tez, Chiron, Kepler, and Oozie [36]. Hi-WAY uses Hadoop 2.0 as the underlying system for managing distributed computational resources and supports the workflows extracted by the Galaxy workflow management system and DAX and Cuneiform workflow languages.

The advantages of Hi-WAY over other workflow management systems include:

- Support of multiple workflow languages
- Iterative workflows
- Improved efficiency with adaptive scheduling
- Scalability

For the evaluation, workflows were processed using both the WSH and HEFT Java code [28] on an HP DL 580 G8 server equipped with an Xeon E-7 4890-Intel processor operating at 2.8 GHz per core. The Hadoop platform

Fig. 8 The job scheduling algorithm

```
1: procedure job Scheduling (WF, job_priority_queue,  Clus_set)
2: Definition of Argumants{
3:            list-of-nodes → nodes that compare with each other to find the best nodes (
              the  criterion for selecting is minimum Finish Time).
4:            AFT_job→ Actual Finish Time for job
5:            node_A→ all nodes that have executed a job
6:            node_In→ a node that hasn't executed a job
7:            EST_job → The earliest start time
8:            EFT_job → the earliest finish time
9:            node_m→ the node that minimizes job_EFT }
10: while there are unscheduled jobs in the job_priority_queue do
11:   Select the first job_j from the job_priority_queue for scheduling;
12:   for each clus_i in the job_j_ Clus_sort  do
13:     if (all node in clus_i have executed at least one job){
14:              for each node_j in the clus_i () do
15:               list-of-nodes.add(node_j);
16:              end for
17:       else {
18:               list-of-nodes.add(node_A);
19:               list-of-nodes.add(node_In);
20:               break;
21:         }
22:      end for
23:   for each node_i in the list-of-nodes do
24:     Compute the value of job_EST[job,node_i] and job_EFT[job, node_i];
25:   end for
26:   Assign job to the node_m, which minimizes EFT_job;
27:   Set AFT_job = EFT_job;
28:   if (node_m= the last node in list-of-nodes)
29:     Add this node(node_m) to nodes in clus_i that have executed a job;
30:     if (node_m=last node of the last clus in  job_Clus_sort){
31:          for each clus_i in the clus_set() do
32:              All nodes are ready to run a job;
33:          end for
34:  list-of-nodes.clear(); // clear the whole items in list-of-nodes.
35:  Remove the job_j from the job_priority_queue;
36:  end while
37:  end procedure
```

was installed on virtual machines created on the server. To construct various computational clusters, we utilized virtual machines, each containing three slave computing nodes. Table 5 presents the specifications of a representative slave node within these clusters. To form four clusters, twelve nodes were designated as slaves, with one node serving as the master. Physical machines were employed to evaluate the algorithm, and the network structure, encompassing the clusters and physical machines, is detailed in Table 6. Specifications for other clusters in different experiments are not included. Hadoop was installed on all physical machines. The master node was configured with a dual-core processor and 2048 megabytes of RAM. Each subcluster comprised three computing nodes within the clusters, composed of virtual machines.

For the master node, a virtual processor (vCPU) was defined, and its memory was set equal to the maximum computational resources of the nodes. The scheduling operation was performed for many workflows with different numbers of jobs using the proposed scheduling algorithm. The output data of Hadoop from Linux machines were used as the statistical population. The results were compared with the results of HEFT. *Socket* and *Core* show the number of processors and cores in virtual machines, respectively.

5.1 Analysis of Performance

The measures used for comparing the performance of HEFT and WSH, i.e., makespan, scheduling length ratio, and speedup, were defined as in [33]. The makespan is when a group of Hi-WAY jobs is finished. Thus, the lower the

Table 5 The heterogeneous computational cluster consists of virtual machines

The heterogeneous computational cluster							
Cluster 1		Cluster 2		Cluster 3		Cluster 4	
Resources		Resources		Resources		Resources	
RAM	vCPU	RAM	vCPU	RAM	vCPU	RAM	vCPU
4096 MB	4 Socket 2 Core	2048 MB	2 Socket 2 Core	1024 MB	1Socket 2 Core	512 MB	1Socket 1 Core

Table 6 The heterogeneous computational cluster consists of physical machines

Computational resources	The heterogeneous computational cluster		
	Cluster 1		Cluster 2
	Slave node 2	Slave node 1	Slave node 3
RAM	1024 MB	1024 MB	2048 MB
CPU	2 Core	2 Core	4 Core

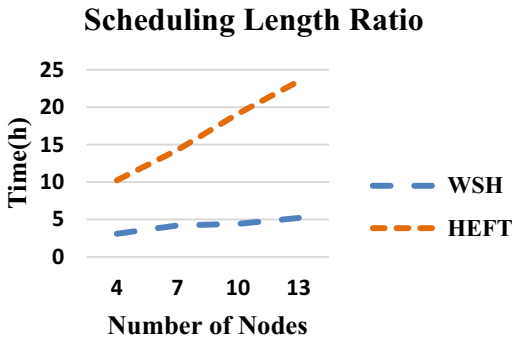

Fig. 9 Scheduling length ratio of the Aviansmall_flu workflow

makespan for a given workflow graph, the better the algorithm performance. Regarding scheduling length ratio (SLR), the better algorithm is with a lower SLR. The measure "speedup" reflects the effect of parallelization and is calculated by dividing the sequential runtime by the parallel runtime (makespan). The effectiveness of these policies was compared according to the different characteristics of graphs and computing nodes. In the experiments, SLR increases and makespan decreases by adding computing nodes to create a scheduler (finding the best computing node for each Hi-WAY job). Figures 10 and 9 show the SLR, and Figs. 12 and 11 show the makespan obtained for Gene2life and Avian-flu_small workflows.

Hi-WAY jobs' allocations to computing nodes after scheduling by HEFT and WSH are presented in Tables 7 and 8.

Figures 14 and 13 show speedup increases with the num-

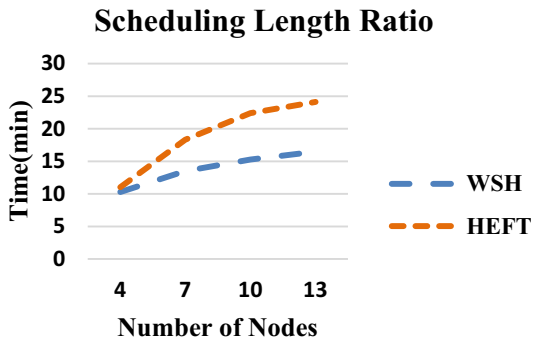

Fig. 10 Scheduling length ratio of the Gene2life workflow

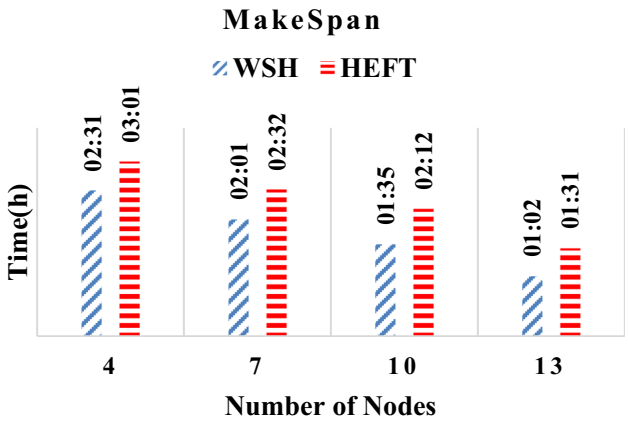

Fig. 11 Makespan of the Gene2life workflow graph

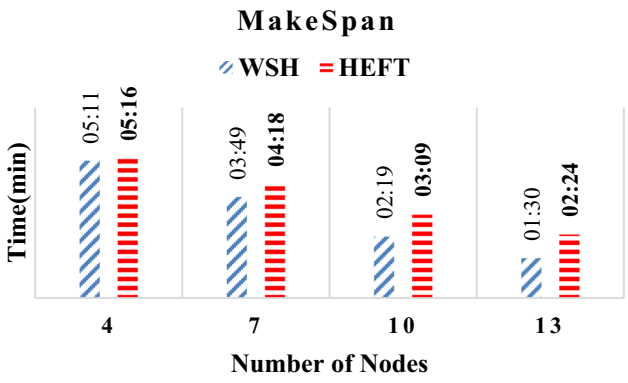

Fig. 12 Makespan of the Aviansmall_flu workflow graph

Table 7 Allocation of Gene2life Hi-WAY job by WSH

Hi-WAY job	Node name	Cluster name
Clustalw1	C1h2datanode2	C1
Protpars	C1h2datanode2	C1
Blast1	C2h2datanode3	C2
Blast2	C1h2datanode1	C1
Drawgram1	C1h2datanode1	C1
Clustalw2	C2h2datanode3	C2
Dnapars	C2h2datanode3	C2
Drawgram2	C2h2datanode3	C2

Table 8 Allocation of Gene2life Hi-WAY job by HEFT

Cluster name	Node name	Hi-WAY job
C2	C2h2datanode3	Blast1
C1	C1h2datanode2	Blast2
C1	C1h2datanode1	Drawgram1
C1	C1h2datanode2	Clustalw1
C2	C2h2datanode3	Dnapars
C2	C2h2datanode3	Drawgram2
C2	C2h2datanode3	Clustalw2
C1	C1h2datanode1	Protpars

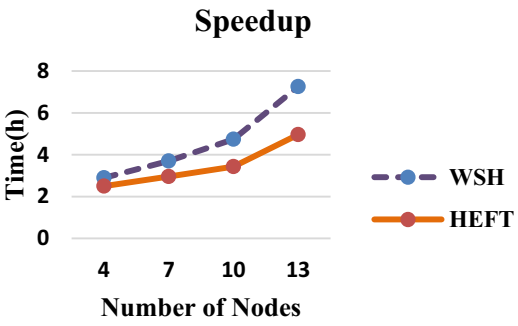

Fig. 13 Speedup in Gene2life workflow graph

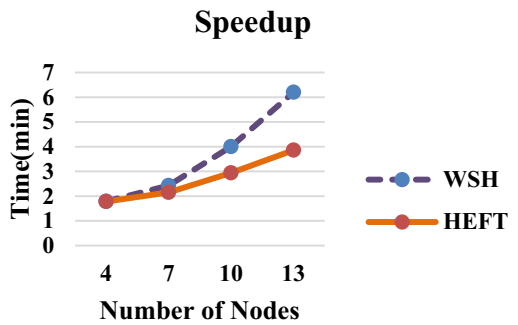

Fig. 14 Speedup in Avianflu_small workflow graph

Table 9 SLR of HEFT and WSH with physical machines

HEFT	WSH	Name of the workflow
00:14:59	00:12:40	gen2life
11:27:00	05:01:00	Avianflu_small

Table 10 SLR of Epigenomics in HEFT and WSH algorithms

HEFT	WSH
31:05:17	16:04:56

Table 11 Scheduler creation time of HEFT and WSH for Epigenomics

HEFT	WSH
17:49:25	06:33:13

Table 12 The runtime of Epigenomics on the schedulers built by HEFT and WSH

HEFT	WSH
13:15:52	09:31:43

ber of computing nodes. Based on these results, in those workflows with a huge number of tasks and where the network is structured, each cluster contains many nodes, WSH performs better than HEFT in terms of SLR, speedup, and runtime.

WSH outperforms HEFT in homogeneous computing nodes. Table 9 shows the SLR obtained for Gene2life and Avianflu_small workflows with three virtual machine computing nodes in each cluster.

The Epigenome center processes DNAs, and the Epigenomics workflow is computing-intensive [14]. Tables 10, 11, and 12 show the SLR, scheduler creation time, and makespan obtained for the Epigenomics workflow.

6 Discussion

Our proposed method for scheduling workflows, utilizing learning tasks and node clustering, has significantly enhanced resource efficiency. While the HEFT algorithm relies on prioritizing tasks without considering the specifics of heterogeneous resources, our approach extracts precise and comprehensive information from each cluster by dividing heterogeneous clusters into homogeneous subclusters. To assess the impact of various structures on the distribution

and efficiency of workflow processing, diverse clusters were employed. This diversity allows us to evaluate the effectiveness and efficiency of resources under different conditions. By varying the number of clusters, the load distribution among nodes is optimized, and resource allocation is facilitated in a more flexible manner. The execution of learning tasks in each cluster retrieves the necessary information from only one node, thereby reducing the need to conduct tests on all nodes. The results of this research demonstrate the superiority of our proposed method compared to HEFT, particularly in heterogeneous and variable conditions. Specifically, a 42% improvement in average scheduling time, a 20% reduction in makespan, and a 37% enhancement in execution speed indicate significant advancements in optimizing resource utilization and task execution time. The scientific applications of this proposed method, as one of the schedulers for scientific and data-driven workflow management systems, are clearly evident, especially in the fields of cloud computing and distributed systems. These systems enable parallel processing and load distribution, which can effectively address large and complex data analysis within shorter time frames. For instance, in research projects such as environmental predictions and disease analysis, these optimizations can provide considerable practical and economic value. Despite the successes achieved, the challenges and limitations of this method are noteworthy. In rapidly changing environments, there is a need for quicker adaptability and more precise configurations. These challenges may impact the overall system performance; thus, future research should focus on addressing these issues.

7 Conclusion and Future Scope

Research corporations and organizations are the owners and sources of massive scientific data today. Using high-level programming languages, researchers can implement the tasks they see fit for their work context and link them with workflow definition languages to produce workflow graphs. Processing workflow graphs involving big data requires powerful distributed workflow systems and scheduling algorithms capable of delivering the desired outputs in a significantly short time by executing multiple tasks on a cluster of homogeneous or heterogeneous computing resources in different distributed systems models. Because of the scheduling policies developed for heterogeneous environments, this research investigated the scheduling of scientific workflow graphs in such settings. Using the introduced algorithm, WSH, workflow graph tasks can be adapted to the nodes' computational capacity throughout scheduling. The experimental results demonstrated the ability of WSH to improve speedup, scheduling length, and makespan. In the future, we will explore the application of advanced optimization algorithms, such as PSO and genetic algorithms. To further evaluate and assess effectiveness, the proposed method will be tested using real-world data and communication costs. Also, communication costs are indeed a significant factor in distributed systems, and the author acknowledges their importance by mentioning the intent to include them in future evaluations.

Acknowledgements "Not applicable."

Authors' contributions All authors contributed equally to this manuscript.

Funding No funding was received.

Declarations

Competing interests There is no conflict of interest.

References

1. J. Bader, L. Thamsen, S. Kulagina, J. Will, H. Meyerhenke, and O. Kao. 2021. Tarema: Adaptive resource allocation for scalable scientific workflows in heterogeneous clusters," in *2021 IEEE International Conference on Big Data (Big Data)*, 2021: IEEE, pp. 65–75.
2. Barika, M.; Garg, S.; Chan, A.; Calheiros, R.N.: Scheduling algorithms for efficient execution of stream workflow applications in multicloud environments. IEEE Trans. on Services Comput. **15**(2), 860–875 (2019)
3. Bittencourt, L.F.; Madeira, E.R.M.: HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds. Journal of Internet Services and Applications **2**(3), 207–227 (2011)
4. M. Bux, J. Brandt, C. Witt, J. Dowling, and U. Leser, "Hi-WAY: execution of scientific workflows on Hadoop YARN," in *20th International Conference on Extending Database Technology, EDBT 2017, 21 March 2017 through 24 March 2017*, 2017: OpenProceedings. org, pp. 668–679.
5. Caíno-Lores, S.; Lapin, A.; Carretero, J.; Kropf, P.: Applying big data paradigms to a large scale scientific workflow: Lessons learned and future directions. Futur. Gener. Comput. Syst. **110**, 440–452 (2020)
6. Cha, S.; Wachowicz, M.: Developing a real-time data analytics framework using Hadoop in 2015. IEEE International Congress on Big Data IEEE (2015). https://doi.org/10.1109/BigDataCongress.2015.102
7. Choudhary, A.; Rajak, R.: A novel strategy for deterministic workflow scheduling with load balancing using modified min-min heuristic in cloud computing environment. Cluster Comput. (2024). https://doi.org/10.1007/s10586-024-04307-8
8. S. E. Dashti. 2015. A New Scheduling Method for Workflows on Cloud Computing," *International Journal of Advanced Research in Computer Science*. 6(6)
9. A. Dwivedi, R. Pant, M. Khari, S. Pandey, L. Mohan, and M. Pande, "E-governance and big data framework for e-governance and use of sentiment analysis," in *International Conference on Advances in Engineering Science Management & Technology (ICAESMT)-2019, Uttaranchal University, Dehradun, India*, 2019.

10. M. R. Girgis, T. M. Mahmoud, and H. M. Azzam. 2024. GA-based QOS-aware workflow scheduling of deadline tasks in grid computing. *Knowledge and Information Systems,* pp. 1–26, 2024.
11. Hanani, A.; Rahmani, A.M.; Sahafi, A.: A multi-parameter scheduling method of dynamic workloads for big data calculation in cloud computing. J. Supercomput. (2017). https://doi.org/10.1007/s11227-017-2050-6
12. Hanen, C.: Study of a NP-hard cyclic scheduling problem: The recurrent job-shop. Eur. J. Oper. Res. **72**(1), 82–101 (1994)
13. Jafarnejad Ghomi, E.; Masoud Rahmani, A.; Nasih Qader, N.: Load-balancing algorithms in cloud computing. J. Network Computer Appl. (2017). https://doi.org/10.1016/j.jnca.2017.04.007
14. Juve, G.; Chervenak, A.; Deelman, E.; Bharathi, S.; Mehta, G.; Vahi, K.: Characterizing and profiling scientific workflows. Futur. Gener. Comput. Syst. **29**(3), 682–692 (2013)
15. M. Khari, M. Kumar, and Vaishali, "Comprehensive study of cloud computing and related security issues," in *Big Data Analytics: Proceedings of CSI 2015*, 2018: Springer, pp. 699–707.
16. J. Koo, I. F. Siddiqui, B. S. Chowdhry, and N. M. F. Qureshi. 2022. Sahws: Iot-enabled workflow scheduler for next-generation hadoop cluster," in *2022 Global Conference on Wireless and Optical Technologies (GCWOT)* IEEE, pp. 1–4.
17. Kouanou, A.T.; Tchiotsop, D.; Kengne, R.; Zephirin, D.T.; Armele, N.M.A.; Tchinda, R.: An optimal big data workflow for biomedical image analysis. Informatics in Medicine Unlocked **11**, 68–74 (2018)
18. K. Krish, A. Anwar, and A. R. Butt. 2014. [phi] sched: A heterogeneity-aware hadoop workflow scheduler," in *Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2014 IEEE 22nd International Symposium on*, 2014: IEEE, pp. 255–264.
19. F. Lehmann, J. Bader, F. Tschirpke, L. Thamsen, and U. Leser. 2023. How workflow engines should talk to resource managers: A proposal for a common workflow scheduling interface, in *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2023: IEEE, pp. 166–179.
20. S. Li *et al.* 2014. Woha: Deadline-aware map-reduce workflow scheduling framework over hadoop clusters," in *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, 2014: IEEE, pp. 93–103.
21. Li, Z., et al.: A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. Futur. Gener. Comput. Syst. **65**, 140–152 (2016)
22. Liu, Q.; Ma, T.; Li, J.; Shen, W.: Workflow Task Scheduling Algorithm Based on IFCM and IACO. In: International Conference on Cloud Computing and Security, pp. 377–388. Springer (2018)
23. Meshkati, J.; Safi-Esfahani, F.: Energy-aware resource utilization based on particle swarm optimization and artificial bee colony algorithms in cloud computing. J. Supercomput. **75**(5), 2455–2496 (2019)
24. Mikram, H.; El Kafhali, S.; Saadi, Y.: HEPGA: A new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment. Simul. Modell. Pract. Theory (2023). https://doi.org/10.1016/j.simpat.2023.102864
25. Mohammadzadeh, A.; Masdari, M.: Scientific workflow scheduling in multi-cloud computing using a hybrid multi-objective optimization algorithm. J. Ambient Intell. Human. Comput. (2023). https://doi.org/10.1007/s12652-021-03482-5
26. Muthuramalingam, S., Bharathi, A., Rakesh Kumar, S., Gayathri, N., Sathiyaraj, R., & Balamurugan, B. (2019). IoT based intelligent transportation system (IoT-ITS) for global perspective: A case study. *Internet of things and big data analytics for smart generation*, 279-300.
27. Polo, J., et al.: Resource-aware adaptive scheduling for mapreduce clusters. In: ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing, pp. 187–207. Springer (2011)
28. Raj, P.; Poongodi, T.; Balusamy, B.; Khari, M.: The internet of things and big data analytics: integrated platforms and industry use cases. CRC Press (2020). https://doi.org/10.1201/9781003036739
29. L. Ramakrishnan and D. Gannon. 2008. A survey of distributed workflow characteristics and resource requirements. *Indiana University,* pp. 1–23, 2008.
30. Rizvi, N.; Ramesh, D.; Rao, P.S.; Mondal, K.: Intelligent salp swarm scheduler with fitness based quasi-reflection method for scientific workflows in hybrid cloud-fog environment. IEEE Trans. Autom. Sci. Eng. **20**(2), 862–877 (2022)
31. B. Saha, H. Shah, S. Seth, G. Vijayaraghavan, A. Murthy, and C. Curino. 2015. Apache tez: A unifying framework for modeling and building data processing applications. in *Proceedings of the 2015 ACM SIGMOD international conference on Management of Data*, 2015: ACM, pp. 1357–1369.
32. R. Sumbaly, J. Kreps, and S. Shah. 2013. The big data ecosystem at linkedin. in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013: ACM, pp. 1125–1134.
33. Tang, Z.; Liu, M.; Ammar, A.; Li, K.; Li, K.: An optimized MapReduce workflow scheduling algorithm for heterogeneous computing. J. Supercomput. **72**(6), 2059–2079 (2016)
34. Topcuoglu, H.; Hariri, S.; Wu, M.-Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
35. G. Turkington, *Hadoop Beginner's Guide*. Packt Publishing Ltd, 2013.
36. J. Wang, D. Crawl, and I. Altintas. 2009. Kepler+ Hadoop: a general architecture facilitating data-intensive applications in scientific workflow systems," In: Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science. 1–8.
37. Wang, S.; Duan, Y.; Lei, Y.; Du, P.; Wang, Y.: Electricity-cost-aware multi-workflow scheduling in heterogeneous cloud. Computing (2024). https://doi.org/10.1007/s00607-024-01264-3
38. J. Xue, T. Wang, and P. Cai. 2023. Towards Efficient Workflow Scheduling Over Yarn Cluster Using Deep Reinforcement Learning," in *GLOBECOM 2023–2023 IEEE Global Communications Conference*, 2023: IEEE, pp. 473–478.
39. Yang, L.; Xia, Y.; Zhang, X.; Ye, L.; Zhan, Y.: Classification-based diverse workflows scheduling in clouds. IEEE Trans. on Autom. Sci. Eng. **21**(1), 630–641 (2022)