Received 30 May 2022, accepted 23 June 2022, date of publication 1 July 2022, date of current version 11 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3187704

RESEARCH ARTICLE

A Unified Mechanism for Cloud Scheduling of Scientific Workflows

ALI KAMRAN[1], UMAR FAROOQ[2], IHSAN RABBI[2], KASHIF ZIA[3], MUHAMMAD ASSAM[4], HADEEL ALSOLAI[5], AND FAHD N. AL-WESABI[6]

[1]Department of Physical and Numerical Sciences, Qurtuba University of Science and Information Technology, Dera Ismail Khan 29050, Pakistan
[2]Department of Computer Science, University of Science & Technology Bannu, Bannu 28100, Pakistan
[3]Faculty of Computing and Information Technology, Sohar University, Sohar 311, Oman
[4]Department of Software Engineering, University of Science and Technology Bannu, Bannu 28100, Pakistan
[5]Department of Information Systems, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, Riyadh 11671, Saudi Arabia
[6]Department of Computer Science, College of Science and Art at Mahayil, King Khalid University, Abha 62529, Saudi Arabia

Corresponding author: Ihsan Rabbi (ihsanrabbi@gmail.com)

This work was supported in part by the Deanship of Scientific Research at King Khalid University through the Large Groups Project under grant number 18/43; and in part by the Princess Nourah Bint Abdulrahman University Researchers Supporting Project under grant number PNURSP2022R303 of Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia.

ABSTRACT Scheduling plays a vital role in the efficient utilization of the available resources in clouds. This paper investigates the capabilities of the current scheduling algorithms of WorkFlowSim framework for processing scientific workflows. These investigations used four different sizes of workloads each, for, five well-known workflows. It was revealed that none of the existing algorithms is capable of efficiently executing all the four sizes of workload for the complete set of workflows. Different algorithms performed better, when they were applied to various workloads of a particular workflow. This fact was used in developing an improved unified mechanism, which is capable of using an existing algorithm that performed well in the past, against the given workload. Evaluation results showed that the proposed mechanism improved over the existing algorithms for 4 out of 5 workflows (Epigenomics, Inspiral, Cyber Shake, and Montage), when tested against an aggregated load of all sizes, in terms of simulation time. For the workflow named SIPHT, however, it responded exactly the same as Max-Min algorithm. The minimum and maximum improvements, against the existing best and worst algorithms, in percentage, for Epigenomics, Inspiral, SIPHT, Cyber Shake and Montage were 16-63, 30-68, 0-69, 30-68, and 9-71 in corresponding order. This work has an additional overhead in terms of a dedicated module to find and store algorithmic performance. It is, however, required once and, thus, the increase in execution time might be marginal. The future work intends to check the impact of compute time towards optimization parameters such as makespan, pricing and deadlines.

INDEX TERMS Cloud computing, scheduling, scientific workflows, WorkFlowSim, CloudSim.

I. INTRODUCTION

Cloud Computing (CC) is a platform, which uses Internet for the provision of data and compute resources based on demand. It enables ubiquitous access to a large set of reconfigurable resources such as servers, computer networks and services [1]. The resources are provided and released rapidly with a little managerial effort. CC and storage solutions provide users and enterprises with various capabilities to store and process their data in either privately owned or third-party data centers [2]. It allows the companies to avoid spending on developing their infrastructure, thus, enabling organizations to concentrate on managing their core businesses [3]. Further, it helps the organizations to quickly deploy their applications, whose maintenance requires very little effort. The resources in clouds are adjusted rapidly, thus, enabling organizations to meet the unpredictable and fluctuating demands [3]. The factors behind the tremendous growth of CC include not only the inexpensive computers, storage devices and efficient computer networks but also machine virtualization, service oriented architectures, and utility based computing [4]. CC has attracted much attention from different communities for

The associate editor coordinating the review of this manuscript and approving it for publication was Agostino Forestiero.

developing not only compute and storage environments but also different cloud based applications. The huge popularity of these environments is due to the fact that providing very own information technology, compute, and storage services need massive investments [5].

Clouds require very limited interaction between the users and service providers [6]. They offer a broad network access and provide resources on demand. It develops a resource pool and provides rapid elasticity. It uses metering facility to provide a measured service [7]. The services offered by the service providers in CC select an appropriate model from a wide range of models available today. However, the three basic models are called, Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [7]. The deployment of applications on the clouds use the SaaS model. To facilitate the applications running on clouds, the PaaS model provides the platforms used by the applications including various tools, languages and libraries, which are supported by service providers. The infrastructural entities such as store, compute and network are offered using the IaaS model [8], [9]. Cloud deployment could be private, community, public or hybrid in nature. However, public clouds are the most favorable choice [10]. Commercial cloud environments such as Microsoft's Azure platform [11] and Google App Engine [12] are serving the customers with varied set of requirements.

Clouds are experiencing promising growth, however, CC has a number of issues that must be addressed for making its services user friendly and reliable [13], [14]. Scheduling is an extremely challenging task, which believes in the efficient utilization of resources for an improved end user experience. In general, it follows a set of rules to efficiently execute the given set of sequential tasks on a computer system. However, scheduling in CC is more difficult as it assigns the tasks (such as scientific workflows) submitted by various users to the available resources in such a way that the total response time is reduced. This overall response is called makespan, which represents the overall time from submission of task to getting response after the successful computation. Compute time is a major factor contributing towards makespan and its reduction will reduce the overall response time.

This paper investigates the current scheduling algorithms, employed in WorkFlowSim (WFSim) framework, in terms of simulation timings and, then, proposes an improved unified mechanism to overcome the major issue stating that none of them is capable of efficiently executing workloads of different sizes for a particular workflow. The rest of this paper is organized as follows. Section II provides the background information especially the scientific workflows, WFSim framework and the scheduling mechanisms found in the literature with an emphasis on those incorporated in WFSim. Investigation results are presented in section III. Section IV presents the problem statement, motivation and goals of this work. Section V describes the proposed mechanism. It, then, provides the evaluation results, which are used to compare the proposed mechanism with existing methods. The space time complexity of the proposed method in presented in section VI. Conclusion and future work is presented in section VII.

II. BACKGROUND AND RELATED WORK

This section describes a number of well-known scientific workflows. It explores the WFSim framework and the scheduling methods both heuristic and meta-heuristic in nature with an emphasis on those employed in WFSim to facilitate the execution of workflows on clouds.

A. SCIENTIFIC WORKFLOWS

A scientific workflow is a logical sequence of tasks constituting an application service. Generally, scientific workflows are represented using Directed Acyclic Graphs (DAGs). To evaluate different aspects of CC such as clustering and scheduling, scientific workflows of different domains are used as a benchmark in the literature. A brief description of the workflows used in this work are presented below:

1. **Montage**: is an astronomical application, which is used to develop large mosaic images of the sky [15]–[17].
2. **Inspiral**: is an application of physics domain, which is used to analyze the data obtained from the Laser Interferometer Gravitational Wave Observatory (LIGO) experiment. The data, it uses to search gravitational waves signatures is obtained using scalable interferometers [16], [17].
3. **CyberShake**: is an application falling in the seismology domain, which is used to develop seismic hazard curves for geographical sites in southern California [16], [17].
4. **Epigenomics**: is a workflow exhibiting data parallelism. It takes multi-lanes DNA sequences generated by solexa machines. The data, thus, received is, then, converted in to a form suitable for mapping [16], [17].
5. **SIPHT**: is a workflow implementing a search service and it is developed to find out small un-translated RNAs (sRNAs), which are capable to regulate processes such as secretion and virulence in bacteria. A number of programs are involved in predicting and annotating the sRNAs encoding patterns [18], [16], [17].

B. WORKFLOWSIM

Simulation studies are usually conducted to measure the effectiveness of different aspects of clouds [19]. CloudSim (CSim) [20] emerged as one of the best simulation environments for clouds. However, it lacks the capability to process scientific workflows. Similarly, it does not include support for the overhead involved in failures and heterogeneity. The optimization techniques such as clustering is also missing in CSim.

To incorporate these features, WFSim [19] was developed as an extension to CSim, by adding an additional layer for managing workflows in it. Researchers can easily extend the basic capabilities of WFSim. They can also check its behavior by conducting tests for different scenarios and configurations regarding workflows. WFSim [19] is open source in nature and it models scientific workflows in terms of DAGs. The basic features, it supports include modeling delays, handling node failures and the implementation of task scheduling for workflows. It includes dynamic as well as static scheduling techniques such as Min-Min and Heterogeneous Earliest Finish Time (HEFT).

WFSim is based on the WorkFlow Management System (WMS) model, which is similar to the well-known WMS called Pegasus [13]. It includes modules related to workflows namely a mapper, an engine, a scheduler and a partitioner. It also includes a clustering engine and a provenance collector. The mapper converts the abstract workflows to executable workflows. The workflow engine manages the flow of data and dependencies between the data. The scheduler assigns resources to jobs while the partitioner splits a workflow in different sub-tasks. The clustering engine is used to make large jobs by combining the small tasks. The module named, provenance collector, is used for tracing the execution of tasks by looking into the historical data [19]. Failure support is available for both task and job levels in WFSim. The WFSim component called, Failure Generator, introduces job or task failures randomly at the compute nodes. The component called, Failure Monitor, maintains the complete record of failures. This information is, then, provided to WMS for re-scheduling.

C. WORKFLOW SCHEDULING MECHANISMS

The schedulers for workflows ensure that the tasks are executed in the correct order. When a workflow is submitted to the scheduler/broker, it invokes the scheduling algorithm, which decides about the cloud and execute node on the cloud for processing the workflow. The scheduler takes a DAG (representing the workflow), the processing capabilities of resources and channel capacity of the communication link, as input. The information in DAG is used to estimate the execution time and make-span for the specific workload size of a particular workflow. Scheduling could be static or dynamic when applied to scientific workflows. To manage virtualized environments in existing WMS systems, dynamic schedulers are used to cope with the randomness of workloads and resources. However, static scheduling is highly recommended, when it is possible to calculate good execution timings for the tasks, and resource availability time and the beginning time for a workflow are known in advance. This is because static scheduling considers the workflow structure in the scheduling process [21].

According to Roger [22], clouds are not great for traditional usage, however, they have some amazing uses. They are costlier than traditional servers for continuous use. However, need based usage of clouds that adopt a pay as you use model has an advantage over traditional servers [22]–[24]. They greatly help organization to concentrate on improving their operational capabilities by eliminating the need for huge investment in developing and subsequently maintaining a dedicated computational infrastructure. Compute intensive applications such as scientific workflows believe in dynamic scheduling as their compute requirements do change with time. This is witnessed by many techniques and studies proposed in recent years [23]–[28].

Scheduling algorithms could use heuristic and meta-heuristic approaches and, therefore, two well-known categories of the heuristic and meta-heuristic algorithms do exist in the literature. The traditional and well known heuristic algorithms include First Come First Serve (FCFS) [29], Round Robin (RR) [29], [30], Minimum Completion Time (MCT) [31]–[33], Min-Min [34], Max-Min [35], Heterogeneous Earliest Finish Time (HEFT) [34], [36], Dynamic HEFT (DHEFT) [37] and Data Aware [36]. The literature exists recently proposed extended versions of these algorithms such as those presented in [38] and [39]. On the other hand, many meta-heuristic methods are proposed in the literature. According to Singh *et al.* [40], they include ant colony optimization, particle swarm optimization, genetic algorithm, cat swarm optimization and artificial bee colony based optimiztion. Ramathilagam and Kandasamy [25] and Toussi and Mahmoud [24] presented novel optimization techniques for workflow scheduling in clouds. The latter technique (presented in [24]) used divide and conquer approach to achieve deadline constrained cost optimization. A flexible deadline driven resource management algorithm for multiple workflows is presented in [26]. Similarly, a bi-objective multi-workflow cloud scheduling mechanism based on re-enforcement learning is presented in [27] while an energy aware multi-workflow scheduling for hybrid clouds is presented in [28].

This work uses WFSim environment to investigate the existing methods employed for scientific workflows. We believe that the recently developed methods are not only complex as compared to basic algorithms but their source code is unavailable to experiment with. Furthermore, we do not develop a new algorithm but proposes an integrated environment which uses existing algorithms based on an identified fact during this work that current algorithms employed in WFSim behave differently against different sizes of a given workflow. Therefore, we currently investigate the algorithms employed in WFSim – the tool used in this research work. The proposed mechanism is made scalable using a flexible structure for storage along with a dedicated module for storing and, then, using additional algorithms, in future.

WFSim framework provides a number of static and dynamic scheduling algorithms for handling workflows. The mechanisms, investigated in this work, are briefly explained below:

1. **First Come First Serve (FCFS):** believes in smallest wait timings and, thus, the resource, which comes first is selected for the task assignment. It is a simple non preemptive strategy. FCFS has small turnaround and response time. However, the jobs smallest in size but arriving late might wait for longer to get processed.

CSim assigns Virtual Machines (VMs) to hosts using FCFS strategy [29].

2. **Round Robin (RR):** is a preemptive strategy, which believes in fairness. The jobs are stored in a circular queue and they are processed in turn. They are all given the same time slice for execution. When a job is not finished in the currently assigned turn, it waits to resume its execution in the next turn. The RR strategy achieves fairness by giving, each job, an equal share in CPU time [29]. The longest jobs, however, need huge amount of time to complete. To avoid this issue, job prioritization is introduced in scheduling. Round Robin (RR) strategy is also used by CSim for the internal assignment of resources to jobs [30].
3. **Minimum Completion Time (MCT):** allocates random tasks to a VM, which offers minimum completion time based on its already assigned workload [31], [32]. It uses the processor speed and current workload of a VM to determine, if it offers the minimum completion time. However, the resources in MCT might be poorly utilized as this strategy might select resources for the execution of tasks, which do not process them in minimum time [33].
4. **Min-Min:** dispatches a given task to the VM, which is capable of processing the task in the lowest possible time [34]. In simple words, it adopts the MCT strategy, which randomly assigns tasks to the resources in such a manner that the tasks must be processed in the least possible time. Min-Min schedules the tasks based on priorities, which are assigned highest to the shortest jobs and lowest to the longest jobs. Min-Min might suffer from significant workload imbalance. Similarly, the longest jobs in Min-Min strategy might also suffer.
5. **Max-Min:** strategy allocates the highest priority to the job, which has the longest processing time and lowest to the job having the least processing time. In each assignment step, a task with highest priority (the one with maximum completion time) is allocated to a resource, which is completed in the lowest time. Since, the longer jobs are given preference over the smaller tasks, the smaller jobs might get delayed [35].
6. **Heterogeneous Earliest Finish Time (HEFT)**: is a heuristic based scheduling method, which works in two stages [34]. The former stage prioritizes the tasks by assigning them upward ranks. The latter stage, then, submits these tasks to the processing nodes using the priorities calculated in the former stage. The upward rank of a task is the expected distance between the current and final tasks in the computation. The input to the HEFT algorithm includes a set of tasks, a set of processing nodes, estimated execution time of each task on every available compute node and time needed to provide the response of a job to the complete set of compute nodes [36].
7. **Dynamic HEFT (DHEFT):** is the dynamic version of HEFT algorithm, which had limitations while executing scientific workflows. The HEFT schedules the tasks in the planning phase, which are, then, allocated to selected resources for processing. If a resource is assigned to a job, which is processing an already assigned job, then, the new job has to wait until a resource finishes that job. DHEFT was developed to solve these issues by equipping it with an efficient and dynamic scheduler for more effective assignment of tasks to the compute nodes [37].
8. **The Data aware algorithm**: included in WFSim belongs to the category of algorithms, which are generally used for solving applications believing in data parallelism. The job, which requires most of the data provided as input is assigned to the selected VM. The algorithm is localized in nature and it, therefore, takes advantage of efficient data distribution at local sites [36].

III. INVESTIGATIONS

The cloud users for workflow scheduling usually target optimization constraints such as deadline, cost, makespan and latency. These constraints generally depend on both computation and communication costs [24], [39], [40]. The proposed work is concentrating on the computation aspect, which covers a major portion towards latency, deadlines as well as the prices paid to the cloud. For this reason, we investigate and discuss the performance of exiting scheduling mechanisms employed in WFSim for executing scientific workflows over a pre-defined cloud environment in terms of simulation time. The future work of this study includes checking the impact of proposed method on the above mentioned parameters in terms of its comparison with exiting mechanisms [24], [39], [40].

A. INVESTIGATION ENVIRONMENT

For investigation and evaluation purposes, this work used a system with a 2.6GHz quad-core processor and a 4GB primary memory, which was running Windows 8.1 operating system. It used version 1.0 of the WFSim framework and version 8.1 of NetBeans IDE.

The cloud environment, this work used, had one data center with five VMs. Each VM had a single processing core. The processing speed and the primary memory of each VM was 1000 MIPS and 512MB RAM correspondingly. They were capable of supporting image of sizes up-to 10000MB.

B. WORKFLOWS AND SCHEDULERS

This work used five workflows namely the Epigenomics, Inspiral, Sipht, CyberShake, and Montage, which were discussed earlier in section II(A), for investigation and evaluation purposes. Each workflow was investigated in terms of four different workload sizes where the smallest jobs were having 24-30 nodes, followed by the jobs having sizes from 46-50, and 100 nodes respectively. The longest jobs were comprised of 997-1000 nodes.

For investigation and comparison purposes, all the 8 scheduling algorithms included in WFSim framework and described earlier in section II(C) were used in this work.

These schedulers are called FCFS, RR, MCT, Min-Min, Max-Min, HEFT, DHEFT and data aware.

C. STATISTICAL PARAMETER(S)

This work used the parameter, Total Time (Simulated), for investigation and comparison purposes. It records the total simulation time needed by an algorithm to run a particular workload size of a given workflow.

D. INVESTIGATION RESULTS

This work conducted a large number of experiments, where each experiment applied an algorithm to a specific size workload of a workflow. Each experiment was conducted, five times, to get an average value for fair results. For each algorithm, we conducted four different experiments based on four types of workload sizes. Investigation results, calculated in terms of simulation time, of these experiments are presented in Table-1. The best values against an algorithm are highlighted in gray.

The HEFT algorithm outperformed the rest of the algorithms, when it was applied to Epigenomics, in all cases except the workload made of 1000 nodes, where RR performed better. In case of Inspiral, DHEFT had better response for the workloads, smaller in size, especially consisting of 30 and 50 nodes. HEFT, however, performed the best for workload comprised of 100 nodes. The RR for the Inspiral workflow proved to be the most economical algorithm, when they were applied to the workloads made of 1000 nodes.

Results for the workflow, SIPHT, showed that it was the only workflow, in which the Max-Min algorithm was consistent in terms of timings against workloads of all sizes.

Mixed results were achieved for the remaining workflows namely: the CyberShake and Montage. The HEFT algorithm, in case of CyberShake, showed optimal performance for the workloads of sizes containing 50 and 100 nodes. However, the Min-Min algorithm performed well for the workload containing 25 nodes (the smallest possible workload). The RR strategy was the best to process the longest workload of size involving 1000 nodes. HEFT again performed better for the workloads of sizes containing 25 and 100 nodes while processing the Montage workflow. For the workload comprised of 50 nodes, the Min-Min had better response. For the longest job, containing 1000 nodes, however, data aware algorithm outperformed the remaining algorithms.

E. DISCUSSION

The investigations, in this work, resulted-in mixed but interesting results. It was revealed that none of the eight algorithms investigated in this work, was able to have a constant behavior against all the five workflows considered in this work. None of them was, even, consistent in behavior, against various sizes of workloads for a particular workflow except SIPHT. In case of SIPHT, the Max-Min algorithm was throughout consistent. HEFT and DHEFT had the most promising performance in most of the cases but except the SIPHT. The RR algorithm performed well for maximum size workloads of all workflows except SIPHT and Montage. In a single instance (the Montage's workload comprises 50 nodes), the Min-Min algorithm had a marginally better result than the HEFT. The MCT and FCFS strategies failed to compete with other algorithms, however, the response of MCT was better than the FCFS in most of the cases.

IV. PROBLEM STAEMENT, MOTIVATION AND GOALS

This section formally presents the problem statement based on investigations along with the motivation behind the proposed work and goals set for this study.

A. PROBLEM STATEMENT

This work explored a large number of workflow schedulers found in the literature with an emphasis on those included in WFSim. It, then, investigated the WFSim schedulers against various sizes of workloads for different scientific workflows. It was revealed that none of these algorithms is able to provide a constant behavior against different sizes of a scientific workflow. For a given workflow, different algorithms achieved better simulation timings, when they were applied to different sizes of workloads. The advanced algorithms such as HEFT and DHEFT showed a little improvement in terms of performance, however, they introduce considerable amount of additional burden and complexity.

This work, therefore, proposes an integrated scheduling mechanism, which utilizes the existing algorithms based on their previous performances instead of developing a new algorithm. It uses the algorithm, which suits well to execute a particular workload size of a specific workflow. The proposed mechanism provides a dedicated mechanism for finding and storing algorithmic performance of an algorithm for its future use.

B. MOTIVATION AND GOALS

The non-existence of a specific scheduling algorithm and a unified mechanism, which is capable of processing different workload sizes of a workflow in optimal timings motivated us to develop the proposed method. The major goals of this work include investigating the algorithms employed at WFSim, proposing an alternate unified mechanism and, then, evaluating and comparing it with current algorithms in terms of simulation time.

V. THE PROPOSED MECHANISM

This section presents the proposed unified mechanism along with its evaluation. It first describes the structure of storage used to facilitate the proposed mechanism. It, then, presents the proposed mechanism and its evaluation and comparison with existing algorithms.

A. HANDLING THE REQUIRED DATA

Since, the proposed mechanism believes in using the past performance of an algorithm as a feedback against a particular workload size of a given workflow. In this connection, this work uses a simple database for maintaining the performance

TABLE 1. Summary of algorithmic responses to various workloads of different scientific workflows.

Scientific Workflow	Algorithm	Workload Total Simulation Time (in Seconds)			
		24/25/30 Nodes	46/50/60 Nodes	100 Nodes	997/1000 Nodes
Epigenomics	HEFT	8361.11	19257.64	210922.33	1877102.34
	DHEFT	12543.36	33023.77	349545.48	1948216.26
	Data Aware	85363.12	142924.61	247610.59	1898716.07
	FCFS	18157.51	239326.81	846667.89	1323814.78
	MAXMIN	54567.85	44131.21	297458.04	1630892.41
	MCT	176888.96	71148.01	319941.36	3154535.04
	MINMIN	109469.91	27545.28	847494.50	1789800.03
	Round Robin	219560.5	102449.17	380998.19	1137562.96
LIGO Inspiral	HEFT	4517.53	5120.11	8091.66	124474.41
	DHEFT	3854.57	4885.07	9683.89	106975.12
	Data Aware	5780.92	17353.31	31640.51	140087.11
	FCFS	11947.04	35477.18	11080.98	86302.71
	MAXMIN	13509.63	8637.01	12765.03	108410.92
	MCT	6875.81	19500.08	24737.72	104907.81
	MINMIN	9357.49	10318.85	10881.63	189462.47
	Round Robin	12213.12	13888.00	20425.18	53152.62
SIPHT	HEFT	6007.64	7431.56	9265.38	70929.16
	DHEFT	6618.97	7623.85	14529.70	130181.16
	Data Aware	14566.30	17862.72	21130.20	184684.36
	FCFS	7233.74	36733.52	12512.45	146415.03
	MAXMIN	5870.19	6145.97	5169.23	55644.44
	MCT	13392.85	24007.92	13839.88	90322.45
	MINMIN	9344.53	12543.51	15195.41	119622.74
	Round Robin	8839.75	15076.30	52438.35	156696.34
CyberShake	HEFT	574.64	895.98	1387.34	9080.42
	DHEFT	751.72	1135.10	2840.30	12027.09
	Data Aware	706.06	1230.64	3624.31	8521.00
	FCFS	908.40	3381.72	14253.54	10298.52
	MAXMIN	585.60	2556.03	3504.64	12767.23
	MCT	717.99	946.62	14591.54	9684.31
	MINMIN	540.82	1646.08	3808.22	11320.05
	Round Robin	1975.44	2792.42	5385.32	5529.99
Montage	HEFT	100.71	262.36	484.75	6469.66
	DHEFT	149.56	462.79	777.14	6262.96
	Data Aware	8892.52	357.09	611.72	4300.23
	FCFS	1535.96	761.41	652.96	7160.62
	MAXMIN	1748.86	503.65	2103.41	4471.23
	MCT	270.03	295.47	599.67	4436.15
	MINMIN	1309.91	222.18	1409.05	5456.26
	Round Robin	190.45	4745.55	1960.14	10595.95

of an algorithm against a particular workload size of a given workflow. This help in facilitating the proposed mechanism and making it flexible - as it manages all the changes at database level and, thus, avoids changes in the structure.

This database manages algorithmic descriptions, workflows and their corresponding workload sizes, and the best performance of an algorithm against a specific size of a workflow, in terms of simulation time. The conceptual model of this database (the Entity Relationship Diagram) based on a detailed study of requirements and keeping possible extension in mind is presented in Figure 1. It shows the strength of relationships with the help of cardinalities and modalities. However, the properties of basic and associative entities are added to the logical design presented below:

Algorithm (AlgorithmCode, AlgorithmName, Authors, Description)

ScientificWorkFlow (SciWorkFlowCode, SciWorkFlowName, Description)

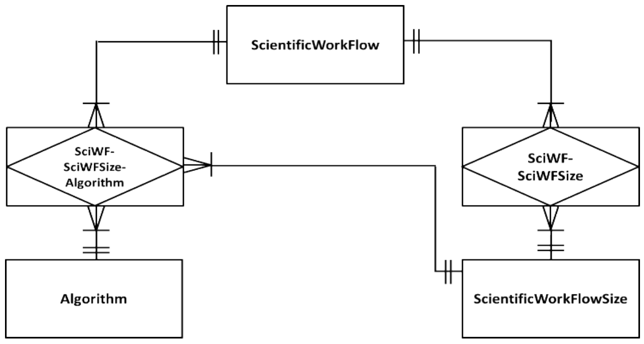


FIGURE 1. The entity relationship model for data requirements of the proposed mechanism.

ScientificWorkFlowSize (SciWorkFlowSizeCode, SciWorkFlowSize, Description)

SciWF-SciWFSize (SciWorkFlowCode, SciWorkFlowSizeCode, Description)

SciWF-SciWFSize-Algorithm(AlgorithmCode, SciWorkFlowCode, SciWorkFlowSizeCode, SimulationTime)

The primary keys in the last two tables, developed for associative entities, in conceptual model (See Figure 1) also serve as foreign keys for their respective base tables. Tables named Algorithm, ScientificWorkFlow and ScientificWorkFlowSize are base tables, which are representing the descriptions of algorithms, scientific workflows and sizes of scientific workflows. The SciWF-SciWFSize table manages the sizes applicable to scientific workflows. The most important information used by the propsoed mechanism (the past optimal performance of an algorithm against a specific size of a workflow) is maintained by the table named: SciWF-SciWFSize-Algorithm.

B. THE PROPOSED UNIFIED APPROACH

Instead of developing a new algorithm, this work develops a unified scheduling mechanism, which selects and, then, uses the best algorithm among those incorporated in the WFSim framework against a particular workload size of a scientific workflow. This mechanism uses the previously known performances, which are explicitly maintained in a database described in section V(A) while selecting the best known algorithm for a particular size of a workflow.

Since, new scheduling algorithms, workflows, and sizes of workflows might be introduced with the passage of time. The proposed method is made capable of adding this information to the database (specifically in tables named: Algorithm, ScientificWorkFlow, ScientificWorkFlowSize and SciWF-SciWFSize) through an explicit dedicated mechanism. Similarly, simulation timings for additional combinations (based on particular algorithm, workflow and size) are determined and stored for future use in a table named: SciWF-SciWFSize-Algorithm. This makes the proposed mechanism very flexible as it always uses up-to-date information. The proposed mechanism along with adding new information is implemented in the form of three algorithms. Algorithm-1 implements the unified scheduling mechanism. Simulation time for an algorithm against a specific size of a workflow is determined and recorded in the database using Algorithm-2. A new scheduling algorithm along with timing information for various sizes of workload for a workflow is stored using Algorithm-3 with the help of Algorithm-2. Algorithm-2 is responsible for determining and storing the performance of an algorithm against a given workflow size of a workflow and it is used by both Algorithm-1 – that realizes the unified aproach and Algorithm-3 – which manages to determine and store performaces of all existing combinations of known workflows and their sizes against a new algorithm. Algorithm-1 uses it when it finds an algorithm whose performance against a particular size of a workflow is not recorded while it is in search of detrmining the best available algorithm. Algorithm-3 uses it when a new algorithm is found in the literature and, thus, the performance of algorithm for all combinations of current workflows and their corresponding workload sizes are determined and stored in database. Algorithm-3 calls Algorithm-2 iteratively until the complete set of combinations are processed.

The working of the proposed mechanism is explained as follows. Algorithm-1 determines the algorithm, which performed better for a given size of workload against a scientific workflow, in terms of simulation time. It requires a DAX file as an input, which contains the workload of a workflow. This file includes the description and size information, which is obtained by the algorithm in sequential order. It first identifies the name of the workload and assigns it to a variable called ScientificWorkLoad. If it encounters a new workflow, which is not in the repository, it is added to the repository. It, then, determines the workload size and assigns it to the variable called, ScientificWorkLoadSize. It is also added to the database, if it encounters a new workload size for a workflow. The proposed technique now collects the details of all known algorithms against the given scientific workflow and a particular workload size. Line 15 to 26 of Algorithm-1 are responsible for determining the best algorithm to process the given workload. This always guarantees the selection of best algorithm as it determines the performance of a newly added workflow or workload size against the current algorithm and records its performance, by calling Algorithm-2 at line 23. Line 24, then, help in reconsidering the performance of the algorithm against workload size of the given workflow. When the loop given in line 15 is terminated, the mechanism must have determined the best algorithm for solving the workload size of given workflow, which is used to process the workload.

Algorithm-2 is sole responsible for finding out the performance of an algorithm in terms of simulation time against a given size of a scientific workflow and storing it in database for references purposes. The proposed unified mechanism presented in Algorithm-1, at line 23, calls this process (Algorithm-2) to calculate the performance of an algorithm, if it is not yet calculated and stored in the database.

Algorithm 1 The Proposed Unified Scheduling Mechanism for Scientific Workflows

```
Require: A ScientificWorkFlow to process; //a DAX file is provided as an input, which comprises of descriptive name and
size of the workflow.
         //Initialisations
1:       char ScientificWorkLoad = "";
2:       int ScientificWorkLoadSize = 0;
3:       char CurrentAlgorithms [] = "";
4:       char AlgorithmToProcessWorkLoad = "";
5:       int AlgorithmProcessTime = ∞;
         //Get information and assign them to corresponding variables and/or data stores
6:       ScientificWorkLoad = Get scientific workflow name from the input file;
         //Store ScientificWorkLoad if not already stored (a new domain) in database
7:       if (ScientificWorkLoad is a new domain) then
8:           add ScientificWorkLoad to the database;
9:       end if
10:      ScientificWorkLoadSize = Get scientific workflow size from the input file;
         //Store ScientificWorkLoadSize if not already stored (a new size) in database
11:      if (ScientificWorkLoadSize is a new size) then
12:          add ScientificWorkLoadSize to the database;
13:      end if
         //Find the algorithm to process the workload, add it to the database if its response is not stored yet.
14:      CurrentAlgorithms [] = Get the current implemented algorithms;
15:      for (int i = 1; i <= CurrentAlgorithms [].Count; i++) do
16:          int CurrentAlgorithmTime;
17:          CurrentAlgorithmTime = Determine the processing time of CurrentAlgorithms [i] against the
             ScientificWorkLoad and ScientificWorkLoadSize;
18:          if (CurrentAlgorithmTime ≠ NULL) then
19:              if (CurrentAlgorithmTime < AlgorithmProcessTime) then
20:                  AlgorithmProcessTime = CurrentAlgorithmTime;
21:              end if
22:          else
23:          Call GetAlgorithmProcessTime (CurrentAlgorithms[i], ScientificWorkLoad,ScientificWorkLoadSize);
24:          decrement i by 1;
25:          end if
26:      end for
27:      Process the ScientificWorkLoad using the algorithm having minimum processing time;
```

Algorithm 2 This Algorithm Finds and Store the Process Time of An Algorithm for a ScientificWorkLoad of a Specific Size

```
Require: char algorithm, char ScientificWorkLoad, int ScientificWorkLoadSize; //This algorithm finds an average simulation
time and store in database
         //Initialisations
1:       int CurrentSimulationTime = 0;
2:       int TotalSimulationTime = 0;
3:       int AverageSimulationTime = 0;
4:       int i = 1;
5:       while(i ≤ 10) do
6:           CurrentSimulationTime = Execute the algorithm for provided size of ScientificWorkLoad;
7:           TotalSimulationTime = TotalSimulationTime + CurrentSimulationTime;
8:       end while
9:       AverageSimulationTime = Round (TotalSimulationTime / 10, 0);
10:      Store AverageSimulationTime for the algorithm against ScientificWorkLoad of ScientificWorkLoadSize in the
         database;
```

Algorithm 3 Logic for Adding a New Algorithm

```
Require: an algorithm; //This procedure takes an algorithm and stores it along with simulation timings for all pairs of
workflows and their workload size.
        //Initializations
1:      char ScientificWorkLoads[] = "";
2:      int ScientificWorkLoadSizes[] = 0;
3:      if (algorithm is not already in the repository) then
4:          add algorithm to the repository;
5:          if (WorkLoads and their corresponding sizes are needed to add to the repository) then
6:              determine the WorkLoads and their corresponding WorkLoadSizes;
7:              add them to the repository;
8:          end if
9:          if (Processing time is needed to generate) then
10:             ScientificWorkLoads[] = Get all existing scientific workflows from the repository;
11:             while (ScientificWorkLoads[] is not exhausted) do
12:                 ScientificWorkLoadSizes[] = Get all possible workload sizes for the current
                    scientific workflow;
13:                 while (ScientificWorkLoadSizes[] is not exhausted) do
                        //Call Algorithm-2 to determine the performance of algorithm and
                        store it in database
14:                     Call GetAlgorithmProcessTime (algorithm, ScientificWorkLoad,
                        ScientificWorkLoadSize);
15:                 end while
16:             end while
17:         end if
18:     else
19:         return: already included in the repository;
20:     end if
```

Algorithm-2 takes three input parameters, which are the current algorithm, ScientificWorkLoad (the workflow), and ScientificWorkLoadSize. To have a fair performance measure, this algorithm calculates a mean value of simulation time based on executing it ten times for the workload size of the given workflow. When this value is calculated, it is recorded for future use as a feedback on performance. It is believed that line 23 in Algorithm-1 might increase the execute time of the proposed algorithm. However, this is added here to make the method robust. This might have very limited use during determining the best known algorithm to execute a given workload, as an independent module is presented in Algorithm-3 to add a new algorithm along with its simulation time for each possible pair of workflow and workload size, to the database.

Algorithm-3 manages new algorithms, when they are found in the literature. It takes a new scheduling algorithm as an input and works as follows. Since, we believe in keeping the simulation timings for an algorithm against all known workflows and their various workload sizes, Algorithm-3 is provided with the proposed integrated mechanism to perform this task. When it finds that the algorithm is not in the repository, it adds it to the database (specifically to the table called: Algorithm). It gets and stores the workflows (interchangeably called Workloads) and their corresponding sizes, if it is needed. Since, it has to process and store simulation time for all combinations of workflows and their corresponding sizes, it collects all scientific workflows in a list called: ScientificWorkloads[]. Algorithm-3, then, iterates this list one by one and collects the corresponding worklod sizes in another list called: ScientificWorkloadSizes[]. The inner while loop defined at line 13, then, iterates through all the sizes one by one and call Algorithm-2 to calculate and store its performance in the database (specifically in table SciWF-SciWFSize-Algorithm). When the inner loop terminates successfully, it suggests that performance of algorithm against all sizes of a given workflow are recorded. When the outer loop located at line 11 terminates, it assures that performance of algorithm for all workloads and their corresponding sizes is recorded. The logic behind Algorithm-2 works exactly the same way described earlier.

C. EVALUATION AND RESULTS

The evaluation of this work uses the same set-up considered earlier for investigations. This includes the system and cloud specifications listed in section III(A), the workflows and schedulers presented in section III(B) and statistical parameter described in section III(C).

To determine the efficiency of the proposed mechanism, this work introduced the cumulative workloads based on

TABLE 2. Comparison of existing and proposed mechanisms for scheduling scientific workflows based on accumulative load.

Scientific Workflow	Algorithm	Cumulative Simulation Time (in Seconds)	Improved by (in %)
Epigenomics	HEFT	2115643.41	35
	DHEFT	2343328.87	41
	Data Aware	2374614.38	42
	FCFS	2427966.99	43
	Max-Min	2027049.51	32
	MCT	3722513.37	63
	Min-Min	2774309.71	50
	RR	1642966.37	16
	The Proposed Method	1376104.04	Not Applicable
LIGO Inspiral	HEFT	142203.70	51
	DHEFT	125398.64	44
	Data Aware	194861.84	64
	FCFS	144807.91	52
	Max-Min	143322.59	51
	MCT	156021.42	55
	Min-Min	220020.44	68
	RR	99678.92	30
	The Proposed Method	69983.92	Not Applicable
SIPHT	HEFT	93633.73	22
	DHEFT	158953.68	54
	Data Aware	238243.57	69
	FCFS	202894.73	64
	Max-Min	72830.02	0
	MCT	140563.09	48
	Min-Min	156706.1	54
	RR	233050.73	69
	The Proposed Method	72830.02	Not Applicable
CyberShake	HEFT	11938.38	30
	DHEFT	16754.21	50
	Data Aware	14082.01	41
	FCFS	28842.17	71
	Max-Min	19413.49	57
	MCT	25940.45	68
	Min-Min	17315.62	52
	RR	15683.16	47
	The Proposed Method	8354.12	Not Applicable
Montage	HEFT	7317.48	30
	DHEFT	7652.44	33
	Data Aware	14161.56	64
	FCFS	10111.02	49
	Max-Min	8827.15	42
	MCT	5601.29	9
	Min-Min	8397.40	39
	RR	17492.09	71
	The Proposed Method	5107.86	Not Applicable

the individual workloads (the original workloads, provided in terms of DAX files), which come bundled with WFSim framework. The individual workloads were used for investigation purposes presented in section III. The cumulative load for a workflow is an integrated load comprises of all individual workloads for the given workflow. These different workloads are assigned to the algorithm all together, which are processed in sequence. The simulation time for a cumulative workload is calculated in the similar fashion but it is now based on timings for all workloads.

Evaluation results for the current algorithms against individual workloads of various scientific were calculated in section III and they were summarized in Table-1. The best response offered by an algorithm against a workload size of a scientific workflow was highlighted in gray.

Table-2 summarizes responses for cumulative loads for the current as well as proposed mechanism, which uses the best algorithm based on historical perspective against workload size of a given workflow. It extends Table-1, where the value of column-3 in Table-2 is almost the same as all responses given in Table-1 for an algorithm against each individual workloads. The response recorded for the proposed method against a workflow, however, is almost the same as the summation of best responses recorded for different algorithms against workloads of various sizes for the scientific workflow. For Example: the HEFT response against the workflow called, Epigenomics, in Table-2 is about 2115643.41Seconds, which could be obtained by adding the values recorded in Column 3 to 6 (individual responses of HEFT for workload of size 25, 50, 100 and 1000) for Epigenomics in Table-1. However, the response recorded for Epigenomics against the proposed method (137610404 Seconds) in Table-2 is the summation of the values given in column 3 to 5 for HEFT (as HEFT performed better for workload sizes of 25, 50, and 100 nodes) and column 6 against RR (as RR performed better for workload size of 100) of Table-1. Similarly, LIGO Inspiral responses are compared as follows. When DHEFT algorithm is used, it produced a response of 125398.64 Seconds (Table-2), which comprises of DHEFT individual responses of 3854.57, 4885.07, 9683.89 and 106975.12 seconds for workload sizes comprises of 25, 50, 100 and 100 nodes (produced in Table-1), in given order. The proposed method on other hand took a total simulation time of 69983.92 Seconds (presented in Table-2). It is almost the same as the combination of individul DHEFT responses of 3854.57 and 4885.07 seconds for workload of size comprises 25 and 50 nodes, 8091.66 seconds HEFT response for 100 nodes and RR response of 53152.62 seconds for a workload size of 1000 nodes. In case of SIPHT, it can be observed that the propsoed method and Max-Min algorithm have both the same simulation time. This is because Max-Min was consistent in performance against all sizes of load and the propsoed method always selected Max-Min based on its perfromance. The rest of the results can be elaborated exactly the same way.

Column-4 of Table-2 is used to compare the proposed method with current methods in terms of cumulative load. Therefore, this column value against the proposed method stands ‘Not Applicable’. This column produces the improvement, in %, offered by the proposed method against the individual algorithm for which the cumulative response time is recorded. In case of Epigenomics, the proposed mechanism improved a minimum of 16% against RR strategy but a maximum of 63% against MCT, in terms of simulation time. In case of Inspiral, it offered 30% improvement at minimum against RR but 68% improvement at maximum against MinMin. In case of SIPHT, the proposed method had exactly the same response as Max-Min (showed 0% improvemnet in Improved by column), however, it improved a maximum of 69% against both data aware and RR strategies. Comparing response for CyberShake revealed that the proposed method improved a minimum 30% against HEFT but a maximum of 71% against FCFS. The proposed method has a minimum and maximum improvement of 9% and 71% against MCT and RR for Montage, in corresponding order.

In short, the proposed method improved over the best existing algorithms for various workflows from 9 to 30% but 63 to 71% against the current worst algorithms based on cumulative load, in terms of simulation timings. The current study improved the response time in 80% cases (4 out of 5 workflows) but remained the same in the remaining 20% cases (1 out of 5 workflows).

VI. SPACE-TIME COMPLEXITY

Since, the proposed mechanism believes in keeping the performance of each algorithm against all possible workload sizes of a workflow and do so for all the workflows. The simple database developed to support the proposed mechanism has five tables to represent algorithms, scientific workflows, scientific workflow sizes, workflow sizes applicable to various workflows, and the performance of algorithms against every workload sizes of all participating workflows. To highlight the space requirements for the current setup, we present here the number of rows in each table. The table Algorithm has only descriptions of 8 algorithms while ScientificWorkFlow has only five (5) entries. There are a total of nine (9) different entries in ScientificWorkFlowSize table, which are 24/25/30, 46/50/60, 100, and 997/1000 nodes. Since, each workflow has currently four different workload sizes, there are only twenty (20) entries for five workflows, in the table named SciWFSciWFSize. The SciWF-SciWFSize-Algorithm is the most populated table of this database, where there are one hundred and sixty (160) entries (8 algorithms × 5 workflows × 4 workload sizes) for algorithmic performance. In case of a new algorithm, there would be about twenty (20) entries in SciWF-SciWFSize-Algorithm table. The overall, space requirements for the proposed algorithm is, thus, negligible and could be implemented with very simple database environments.

Time complexity of the proposed method is performed using asymptotic analysis. Here, it is assumed that in most of the cases, a new algorithm is added using Algorithm-3, which assesses and stores the performance of the algorithm with the help of Algorithm-2. Therefore, Algorithm-1 does not require consulting Algorithm-2 for determining the simulation time required to process a workload of a given size for a scientific workflow by an algorithm. So, Algorithm-1, in this case, of the proposed mechanism is linear in nature. It is important to note that this time is just the time complexity of the proposed mechanism and it does not include the timings required by the scheduling algorithm itself. Algorithm-2, which is used to determine simulation time required by an algorithm to process a given workload size of a workflow is also linear in nature.

In term of complexity, the most expensive algorithm among the three used in this work is Algorithm-3, which determines the simulation time for every workload size of each workflow against an algorithm. The asymptotic analysis of this algorithm, individually, turns to be quadratic in nature. Since, the simulation time which is calculated by Algorithm-2, is an average value, determined, on the basis of 10 times execution of the algorithm for the same pair of workflow and workload size, it adds a third level of nesting to the complexity of Algorithm-3. Thus, its complexity is represented by a polynomial of power 3. However, all the loops involved in the nesting process are executed very limited number of times (particularly, the first, second and third while loop are executed 5, 4 and 10 times) and, thus, the overall effect of the algorithm is not much higher. Furthermore, this investment of time is required once and the actual algorithmic timings for the unified mechanism is linear in nature.

VII. CONCLUSION AND FUTURE WORK

Cloud computing has eliminated the need of huge investments as it offers powerful resources to customers based on a simple pay as you go model. Scheduling is one of the most challenging aspects in cloud, which is explored by many researchers with the basic aim of efficient utilization of resources. Scheduling algorithms play vital role in achieving this goal. This paper investigated the capabilities of existing scheduling algorithms used for scheduling scientific workflows on clouds, which are employed in the WorkFlowSim toolkit – the most widely studied and used simulation environment for simulating workflows on clouds. It, then, developed a unified scheduling mechanism to overcome the limitations of these algorithms.

This work investigated and compared the responses of 8 algorithms against four different workloads of five scientific workflows, in terms of simulation time. The algorithms that were investigated include HEFT, DHEFT, Data Aware, FCFS, MAXMIN, MCT, MINMIN and Round Robin. The five workflows used in this work were Cybershake, Epigenomics, Inspiral, Montage and SIPHT. This work used small to medium to large sized workflows containing 24/25/30, 46/50/60, 100, and 997/1000 nodes. This investigation revealed that no mechanism among those investigated in this work was capable of providing optimal results against all types and sizes of scientific workflows. MAXMIN had a stable response but only against SIPHT. For the remaining workflows, its response was similar to other algorithms. These mixed results inspired us to develop a unified scheduling mechanism, which uses an existing algorithm based on its prior performance for solving a particular workload size of a given workflow. The proposed unified mechanism uses different algorithms for different sizes of the same scientific workflow and does so by using their past history. This is realized with the help of a component of the proposed mechanism that determines the simulation timings of all possible combinations of workflows and their sizes against an algorithm. This makes the proposed mechanism flexible and scalable. This work developed a prototype of the proposed mechanism using WorkFlowSim and evaluated it using cumulative workloads of scientific workflows comprises of their corresponding individual workloads of different sizes. The suggested mechanism was compared with existing algorithms, which showed that the proposed mechanism always selects the best algorithm to run a given workload and, thus, improves the overall simulation time as compared to the results achieved by each algorithm in individual capacity. Evaluation results revealed that the proposed mechanism improved against the existing algorithms for all workflows except SIPHT (4 out of 5). In case of SIPHT, it obtained the same response as MAXMIN. It obtained up to a maximum of 71% improvement over the existing mechanisms.

This work after careful investigations of existing mechanisms proposed a unified mechanism and it was evaluated and compared with current techniques. The following are some of the future insights about this work:

1. This work used a limited set of workflows and algorithms, thus, it would be interesting to use it against workflows of other domains and add more algorithms into it. Similarly, it was evaluated four different sizes and it could be interesting to extend it for using workloads greater than 1000 nodes.
2. The majority of the algorithms investigated in this work fall in the category of heuristic algorithms. In future, the same platform can be used to investigate those algorithms, which fall within the category of metaheuristics such as particle swarm optimization and genetic algorithms.
3. This work used a parameter called 'simulation timings' to compare the current and proposed techniques. In future, it would be interesting to compare them based on processing time. The impact of processing time on total makespan, deadline and pricing would be an interesting point to investigate.
4. The proposed mechanism was implemented using WorkFlowSim toolkit. It would be interesting to extend its implementation to CloudSim framework, so it can accommodate simulating other domains than scientific workflows.

REFERENCES

[1] S. Dhanasekaran and V. Vasudevan, "A dynamic multi-intelligent agent system for enhancing the cloud service negotiation," *Int. J. Appl. Eng. Res.*, vol. 10, no. 43, pp. 30469–30473, 2015.

[2] N. Manikandan, S. Devayani, and M. Divya, "Domain-specific allocation & load balancing in cloud computing using virtual machines," in *Proc. 4th Int. Conf. Trends Electron. Informat. (ICOEI)*, Jun. 2020, pp. 467–472, doi: 10.1109/ICOEI48184.2020.9143006.

[3] M. S. Aksoy and D. Algawiaz, "Knowledge management in the cloud: Benefits and risks," *Int. J. Comput. Appl. Technol. Res.*, vol. 3, no. 11, pp. 718–720, Nov. 2014.

[4] S. Bhardwaj, L. Jain, and S. Jain, "Cloud computing: A study of infrastructure as a service (IAAS)," *Int. J. Eng. Inf. Technol.*, vol. 2, no. 1, pp. 60–63, 2010.

[5] L. Wang, G. V. Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud computing: A perspective study," *New Generat. Comput.*, vol. 28, no. 2, pp. 137–146, 2010.

[6] S. E. Shukri, R. Al-Sayyed, A. Hudaib, and S. Mirjalili, "Enhanced multi-verse optimizer for task scheduling in cloud computing environments," *Expert Syst. Appl.*, vol. 168, Apr. 2021, Art. no. 114230.

[7] H. AlHakami, H. Aldabbas, and T. Alwada'n, "Comparison between cloud and grid computing: Review paper," *Int. J. Cloud Comput., Services Archit.*, vol. 2, no. 4, pp. 1–21, Aug. 2012.

[8] L. K. Arya and A. Verma, "Workflow scheduling algorithms in cloud environment—A survey," in *Proc. Recent Adv. Eng. Comput. Sci. (RAECS)*, Mar. 2014, pp. 1–4.

[9] M. Adebiyi, E. Adeka, F. Oladeji, R. O. Ogundokun, M. O. Arowolo, and A. A. Adebiyi, "Evaluation of load balancing algorithms on overlappiing wireless accesspoints," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 21, no. 2, pp. 892–902, 2021.

[10] T. Kajiyama, M. Jennex, and T. Addo, "To cloud or not to cloud: How risks and threats are affecting cloud adoption decisions," *Inf. Comput. Secur.*, vol. 25, no. 5, pp. 634–659, Nov. 2017.

[11] S. Krishnan, *Programming Windows Azure: Programming the Microsoft Cloud*. Sebastopol, CA, USA: O'Reilly Media, 2010.

[12] S. Abrahao and E. Insfran, "Models@runtime for monitoring cloud services in Google app engine," in *Proc. IEEE World Congr. Services (SERVICES)*, Jun. 2017, pp. 30–35.

[13] R. Khan and A. Mehmood, "Realization of interoperability & portability among open clouds by using agents mobility & intelligence," *Int. J. Multidisciplinary Sci. Eng.*, vol. 3, no. 7, pp. 7–11, 2021.

[14] J. Deng, S. C.-H. Huang, Y. S. Han, and J. H. Deng, "Fault-tolerant and reliable computation in cloud computing," in *Proc. IEEE Globecom Workshops*, Dec. 2010, pp. 1601–1605.

[15] M. Adhikari, T. Amgoth, and S. N. Srirama, "A survey on scheduling strategies for workflows in cloud environment and emerging trends," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 1–36, Jul. 2020.

[16] R. F. D. Silva. (2022). *Deprecated Workflow Generator*. Accessed: Feb. 2022. [Online]. Available: https://confluence.pegasus.isi.edu/display/pegasus/Deprecated+Workflow+Generator

[17] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. F. da Silva, M. Livny, and K. Wenger, "Pegasus, a workflow management system for science automation," *Future Gener. Comput. Syst.*, vol. 46, pp. 17–35, May 2014, doi: 10.1016/j.future.2014.10.008.

[18] C. Chen, J. Liu, Y. Wen, and J. Chen, "Research on workflow scheduling algorithms in the cloud," in *Proc. Int. Workshop Process-Aware Syst.* Berlin, Germany: Springer, 2014, pp. 35–48.

[19] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. IEEE 8th Int. Conf. E-Sci.*, Oct. 2012, pp. 1–8.

[20] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exp.*, vol. 41, no. 1, pp. 23–50, Aug. 2011.

[21] A. V. Boukhanovsky and S. V. Ivanov, "Urgent computing for operational storm surge forecasting in saint-petersburg," *Proc. Comput. Sci.*, vol. 9, pp. 1704–1712, Jan. 2012.

[22] T. Rogers. (2013). *Should You Host on the Amazon Cloud?* Accessed: May 2022. [Online]. Available: http://www.hypergridbusiness.com/2013/11/should-you-host-on-the-amazon-cloud/

[23] Z. Ahmad, A. I. Jehangiri, M. A. Alaanzy, M. Othman, R. Latip, S. K. U. Zaman, and A. I. Umar, "Scientific workflows management and scheduling in cloud computing: Taxonomy, prospects, and challenges," *IEEE Access*, vol. 9, pp. 53491–53508, 2021.

[24] G. K. Toussi and M. Naghibzadeh, "A divide and conquer approach to deadline constrained cost-optimization workflow scheduling for the cloud," *Cluster Comput.*, vol. 24, no. 3, pp. 1711–1733, Sep. 2021.

[25] A. Ramathilagam and K. Vijayalakshmi, "Workflow scheduling in cloud environment using a novel Metaheuristic optimization algorithm," *Int. J. Commun. Syst.*, vol. 34, no. 5, Mar. 2021, Art. no. e4746.

[26] P. Rajasekar and Y. Palanichamy, "A flexible deadline-driven resource provisioning and scheduling algorithm for multiple workflows with VM sharing protocol on WaaS-cloud," *J. Supercomput.*, vol. 78, pp. 8025–8055, Jan. 2022.

[27] H. Li, J. Huang, B. Wang, and Y. Fan, "Weighted double deep *Q*-network based reinforcement learning for bi-objective multi-workflow scheduling in the cloud," *Cluster Comput.*, vol. 25, no. 2, pp. 751–768, Apr. 2022.

[28] H. Li, G. Xu, D. Wang, M. Zhou, Y. Yuan, and A. Alabdulwahab, "Chaotic-nondominated-sorting owl search algorithm for energy-aware multi-workflow scheduling in hybrid clouds," *IEEE Trans. Sustain. Comput.*, early access, Jan. 21, 2022, doi: 10.1109/TSUSC.2022.3144357.

[29] E. Ilavarasan and P. Thambidura, "Low complexity performance effective task scheduling algorithm for heterogeneous computing environments," *J. Comput. Sci.*, vol. 3, no. 2, pp. 94–103, Feb. 2007.

[30] V. Vignesh, K. S. S. Kumar, and N. Jaisankar, "Resource management and scheduling in cloud environment," *Int. J. Sci. Res. Publications*, vol. 3, no. 1, pp. 1–6, 2013.

[31] M. Lavanya, B. Shanthi, and S. Saravanan, "Multi objective task scheduling algorithm based on SLA and processing time suitable for cloud environment," *Comput. Commun.*, vol. 151, pp. 183–195, Feb. 2020.

[32] S. K. Panda and P. K. Jana, "Load balanced task scheduling for cloud computing: A probabilistic approach," *Knowl. Inf. Syst.*, vol. 61, no. 3, pp. 1607–1631, Dec. 2019.

[33] X. Li, Y. Mao, X. Xiao, and Y. Zhuang, "An improved max-min task-scheduling algorithm for elastic cloud," in *Proc. Int. Symp. Comput., Consum. Control (IS3C)*, Jun. 2014, pp. 340–343.

[34] F. Mohammadi, S. Jamali, and M. Bekravi, "Survey on job scheduling algorithms in cloud computing," *Int. J. Emerg. Trends Technol. Comput. Sci.*, vol. 3, no. 2, pp. 151–154, 2014.

[35] S. S. Brar and S. Rao, "Optimizing workflow scheduling using max-min algorithm in cloud environment," *Int. J. Comput. Appl.*, vol. 124, no. 4, pp. 44–49, Aug. 2015.

[36] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.

[37] H. Yazdanpanah, A. Shouraki, and N. Jamali, "Evaluation performance of task scheduling algorithms in heterogeneous environments," *Int. J. Comput. Appl.*, vol. 138, no. 8, pp. 1–9, Mar. 2016.

[38] H. Mahmoud, M. Thabet, M. H. Khafagy, and F. A. Omara, "An efficient load balancing technique for task scheduling in heterogeneous cloud environment," *Cluster Comput.*, vol. 24, no. 4, pp. 3405–3419, Dec. 2021.

[39] M. H. Shirvani and R. N. Talouki, "A novel hybrid heuristic-based list scheduling algorithm in heterogeneous cloud computing environment for makespan optimization," *Parallel Comput.*, vol. 108, Dec. 2021, Art. no. 102828.

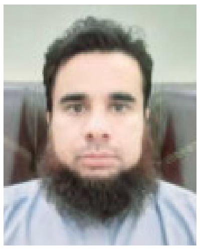
[40] P. Singh, M. Dutta, and N. Aggarwal, "A review of task scheduling based on meta-heuristics approach in cloud computing," *Knowl. Inf. Syst.*, vol. 52, no. 1, pp. 1–51, Apr. 2017.

ALI KAMRAN received the M.S. degree in computer science from the Qurtuba University of Science and Information Technology, Dera Ismail Khan, Khyber Pakhtunkhwa, Pakistan. He is currently working as a Lecturer at the Government Associate College of Commerce, Bhakkar Higher Education Department, Government of the Punjab Pakistan. His research interest includes cloud computing with an emphasis on scheduling.

UMAR FAROOQ received the Ph.D. degree in computer science from the University of East Anglia, Norwich, U.K., in 2012. He is currently working as an Assistant Professor of computer science at the University of Science and Technology Bannu, Pakistan. His research interests include parallel and distributed simulations, virtual worlds, wireless and mobile computing, and grid, cloud, and ubiquitous computing with an emphasis on scalable infrastructures.

IHSAN RABBI received the master's degree in computer science from the University of Peshawar, Pakistan, in 2008, and the Ph.D. degree in computer science from the University of Malakand, Chakdara, Pakistan. He is currently working as an Associate Professor with the Department of Computer Science, University of Science and Technology Bannu, Pakistan. His research interests include augmented reality and virtual reality.

KASHIF ZIA received the master's degree in computer science and the Ph.D. degree in informatics from the University of Linz, Austria, in 2013. He is currently working at Sohar University, Oman, as an Associate Professor. He has active collaboration with the Institute of Pervasive Computing, University of Linz. He is an Engineering Graduate of the University of Engineering and Technology, Lahore, Pakistan, followed by the Master of Computer Science. His research interests include complex adaptive systems, socio-technical systems, agent-based modeling, and complexity science.



MUHAMMAD ASSAM received the B.Sc. degree in software engineering from the University of Engineering and Technology Peshawar, in 2011, and the M.Sc. degree in software engineering from the University of Engineering and Technology Taxila, Pakistan, in 2018. He is currently pursuing the Ph.D. degree in computer science and technology with Zhejiang University, China. He is working as a Lecturer at the Department of Software Engineering, University of Science and Technology Bannu, Pakistan. His research interests include brain–machine interface, medical image processing, machine/deep learning, the Internet of Things (IoT), and computer vision.

HADEEL ALSOLAI is currently an Academic Teacher at the College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University. She has expertise in conducting empirical studies of software engineering techniques (particularly software maintainability, software quality, and open-source systems), along with machine learning techniques (particularly ensemble techniques, data pre-processing, and parameter tuning). Her research and teaching interests include the general area of artificial intelligence and software engineering.

FAHD N. AL-WESABI received the Ph.D. degree in computer science from SRTM University, India, in 2015. He was an Assistant Professor with the Faculty of Computer and Information Technology, Sana'a University, Yemen. Since October 2018, he has been an Assistant Professor with the Computer Science Department, King Khalid University, Saudi Arabia. He is the author of ten books, more than 80 articles, and many funded research projects. His research interests include AI, the IoT, smart cities, machine learning, biomedical, software engineering, applied soft computing, information security, and enterprise systems.