Received 7 August 2022, accepted 18 August 2022, date of publication 29 August 2022, date of current version 9 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3202651

RESEARCH ARTICLE

Levelized Multiple Workflow Allocation Strategy Under Precedence Constraints With Task Merging in IaaS Cloud Environment

FAISAL AHMAD[1], MOHAMMAD SHAHID[2], (Member, IEEE), MAHFOOZ ALAM[3], ZUBAIR ASHRAF[4], MOHAMMAD SAJID[3], KETAN KOTECHA[5], AND GAURAV DHIMAN[6,7,8], (Senior Member, IEEE)

[1]Workday Inc., Pleasanton, CA 94588, USA
[2]Department of Commerce, Aligarh Muslim University, Aligarh 202002, India
[3]Department of Computer Science, Aligarh Muslim University, Aligarh 202002, India
[4]Department of Computer Science and Engineering, Koneru Lakshmaiah University, Vaddeswaram, Andhra Pradesh 522302, India
[5]Symbiosis Institute of Technology, Symbiosis International (Deemed) University, Pune 412115, India
[6]Department of Electrical and Computer Engineering, Lebanese American University, Beirut 1102, Lebanon
[7]University Centre for Research and Development, Department of Computer Science and Engineering, Chandigarh University, Gharuan, Mohali 160036, India
[8]Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun 248002, India

Corresponding authors: Ketan Kotecha (director@sitpune.edu.in) and Mohammad Shahid (mdshahid.cs@gmail.com)

This work was supported in part by the Research Startup Grant from the University Grant Commission (UGC)-Basic Scientific Research (BSR) under Grant F.30-377/2017 (BSR) and Grant 10057/B30528; and in part by the Symbiosis International (Deemed University), Pune, India.

ABSTRACT Cloud Service Providers are speedily becoming the target platform for scientific workflow computations due to the massive possible and flexible pay-as-you-go pricing model. Workflow allocation problem in cloud systems is considered NP-hard. A heterogeneous IaaS cloud could be fully effective if the allocation method provides an efficient mapping between virtual machines (VMs) and workflow applications demanding execution. First, we model multiple workflow allocation problem in the cloud environment. Then, we propose a levelized multiple workflow allocation strategy with task merging (LMWS-TM) to optimize turnaround time for multiple workflow applications in the Infrastructure as a Service (IaaS) cloud environment to achieve better performance. The task merging scheme is incorporated into workflows after partitioning and prior to allocation to reduce inter-task communication share and the total number of depth levels for improving the overall completion time. Moreover, it considers inter-task communication and inter-machine distance for estimating communication cost share among tasks on the schedule generated. Furthermore, the scheme is capable enough to use simple and flexible level attributes to tackle precedence constraints. Afterward, we conducted an experimental study to evaluate LMWS-TM by comparative performance analysis with its peers, namely SLBBS, DLS, and HEFT, on quality of service (QoS) parameters, namely, turnaround time, system utilization, flow time, and response time. The study reveals the superior performance of LMWS-TM among its considered peers in almost all the cases for almost all considered parameters under investigation. Finally, we performed statistical testing to test the significance level using SPSS 20, confirming the hypothesis drawn in the experimental study.

INDEX TERMS Cloud computing, IaaS cloud environment, multiple workflow allocation, task merging, levelized DAG scheduling, turnaround time.

I. INTRODUCTION

Infrastructure as a Service (IaaS) cloud system consists of heterogeneous processing resources interconnected via a high-speed network, capable of matching the requirements of industrial and scientific distributed applications on-demand through collaborative sharing. These systems facilitate sharing of heterogeneous geographically distributed resources in a dynamic environment and self-aggregation depending on cost, availability, capability, performance, and customer

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

demands [1], [2]. Here, the first and foremost concern is allocating compute-intensive resources over workflow applications to optimize the quality of service (QoS) parameters. These facilities are provided on a rental basis and are also referred to as the pay-per-use model. Cloud Service Providers (CSP) features for workflow computations include elasticity, fault tolerance, reliability, and flexible access to virtualized resources.

Workflow allocation in the cloud system has been considered the problem of allocation of workflow tasks on virtual resources with matching workflow's requirements with the effective allocation of parallel executable portion in the workflow over the available hardware parallelism [3], [4]. The allocation of workflow applications with precedence constraints has been considered to be NP-Hard [5]. Over the past years, the workflow allocation problem has been extensively studied, focusing on heterogeneous distributed systems like clusters, cloud, and their successor systems [6], [7]. The workflow applications consist of cooperative tasks modeled using Direct Acyclic Graph (DAG). Here, the terms DAG and workflow are used interchangeably. These applications include web service workflows, scientific workflows, and big data processing workflows, i.e., Map Reduce from Google, Dryad from Microsoft, etc. [8], [9], [10], [11]. The scientific workflow management systems manage the deployment of scientific workflows onto virtualized distributed resources. Currently, an example of scientific workflow technology is the detection of gravitational waves by the LIGO project [12]. Workflow allocation is required to consider the heterogeneity of the computing power of machines and communication edges. An important issue in workflow allocation is how to rank the tasks in the DAG. The task rank is used as its priority in the allocation and computes an order of execution of workflow tasks to preserve the precedence constraints. Once the tasks are ranked, the task to resource assignment can be found to minimize the schedule length or other QoS parameters. On the other hand, the level attribute (lowest level gets the highest priority) is also used to assign priorities to tasks in the workflow to manage precedence orders in execution [3], [4], [13], [14], [15]. Tasks at the same level are parallel executable portions of workflow.

The multiple workflows allocation is used to improve the exploitation of the overall parallelism resulting in optimized parameters [3], [16]. Multiple workflow allocation aims to map a batch of merged workflows on suitable heterogeneous virtual resources with precedence and batch constraints so that the processing can optimize demanded parameters as per the workflow user's specifications. The challenges of multiple workflow allocation in the cloud environment are workload heterogeneity, performance variability, QoS diversity, fairness, and priority [17]. Some multiple application workflows need to be allocated onto the virtual resources. One of the straightforward methods for this is to merge multiple workflows and then execute them like a single workflow allocation. The two allocation policies and four composition approaches for merging have been presented in [18]. The merge-based allocation approaches are able to utilize the static allocation benefits. But multiple workflows are allocated immediately as per their arrival, and results are finally combined in the dynamic environment. Other multiple workflow allocation models use the level attribute to assign priorities, and managing precedence constraints to optimize one or more QoS parameters has also been proposed in the literature [16], [19], [20], [21], [22]. In this case, workflows are grouped according to the depth level, followed by mapping onto resources by exploiting the parallel executable tasks at the same depth level within the batch of DAGs.

This paper proposes a levelized multiple workflow allocation strategy with task merging (LMWS-TM) to optimize turnaround time for multiple workflows with precedence constraints represented by a DAG in the IaaS cloud environment. The turnaround time of a job submitted to a cloud environment is a very important QoS parameter that directly affects customer satisfaction, providers' reputation, energy consumption, monetary cost, and overall business. Lower turnaround time is desirable and results in better performance on other parameters too. The proposed strategy combines the benefits of both widely adopted approaches, such as levelized list-based scheduling and task merging approaches. LMWS-TM is capable enough to use simple and flexible level attributes to tackle precedence constraints. Level attribute provides the order of execution to tasks in the workflows by assigning them execution priorities (highest priorities are for the lowest level and vice versa). Levelized list-based allocation approach minimizes the actual execution time of the workflows submitted for execution. While the task merging scheme is employed to reduce inter-task communication cost share and the total number of depth levels to enhance overall completion time. A task merging scheme is incorporated into the batch of workflows after partitioning and prior to allocation. LMWS-TM considers the inter-task communication and the inter-machine distances to estimate communication costs between any pair of tasks and VMs. The task merging scheme is incorporated in SLBBS [23], and the allocation pattern is also modified. An experimental study has been carried out to measure the effect of the task merging scheme on SLBBS. A comparative analysis has been done with its peers, namely SLBBS, DLS, and HEFT, on various cases for performance evaluation of LMWS-TM on quality of service (QoS) parameters namely, turnaround time, system utilization, flow time, and response time. Further, statistical analysis has been performed to confirm the results and significance level of the simulation study. In summary, LMWS-TM has been proposed for possible improvement on the performance parameters over SLBBS [23]. The proposed strategy has been designed with the following advantages and contributions, listed as follows

- Many methods exploit the parallel executable portion available only at the workflow level. Due to multiple workflows and levelized partition, LMWS-TM exploits parallelism at both workflow and task levels.

- Many models in the literature used ranking methods for preserving the precedence constraints. But the proposed model uses the level attribute for the purpose. The level attribute is the most convenient and flexible way to preserve precedence constraints in workflows.
- The proposed strategy divides the workflows into partitions as accordance to depth level. Slicing all workflows according to depth level minimizes the response time as all workflows start their execution almost in the first depth level. So, LMWS-TM is also quite suited for applications for interactive users.
- The merging phase works as pre-processing on the workflows, making them quite suitable for allocation for possible improvement.
- Further, in the task merging phase, tasks from higher depth levels are shifted and merged with the tasks of lower depth levels. This process will reduce the total number of depth levels in the batch. Consequently, the considered performance parameters are expected to be improved.
- Merging tasks also minimizes the communication cost share on the allocation of the workflows because more tasks are combined into one and allocated on a single suitable virtual machine resulting in the communication share among them being zero. Again, improvement is expected.
- Experimental evaluation and statistical analysis are conducted to evaluate the proposed LMWS-TM strategy.

The rest of the paper is as follows: Section II presents some related work from the domain. Section III explains the proposed model by presenting various notions used, cloud framework, mathematical models, problem formulation with parameter estimation, an algorithm with illustration, motivations, and time complexity. The experimental study to evaluate the comparative performance of LMWS-TM has been presented in Section IV. Also, in Section IV, a statistical analysis has been conducted to test the validity of the hypothesis developed. The paper concludes the work with some future directions in Section V.

II. RELATED WORK

Workflow allocation problem has been considered an NP-hard due to the heterogeneity and dynamic nature of applications, resources, and communication links in an IaaS cloud environment with the requirement of proper security of transferred information and its execution on the distributed resources [5]. Therefore, many workflow allocation methods have been developed to solve the problem in the literature using various approaches. These models can be categorized into a single workflow and multi-workflow allocation models. Further, Single workflow models are classified into static workflow allocation models [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47] and dynamic workflow allocation models [34], [35], [36], [37]. Static workflow allocation is done at compile time. Here, all information regarding the task is known in advance, such as data dependencies among the tasks, execution time, etc. It is beneficial in many research areas such as simulation studies, post-mortem analyses, and designing a system. One of the essential advantages of this approach is that if a schedule is prepared for allocation, one can ensure that all limits will be guaranteed. Dynamic workflow allocation takes the decisions at run time. As soon as new tasks arrive, the scheduler dynamically decides the possibility of allocating these new tasks without exposing the assurances provided. The static single workflow allocation approach can be extended into list scheduling [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [39], clustering [40], [41], [42], [43], [44], [45], [46], [47], etc.

In a list-based heuristic, a list of all tasks is prepared from the given DAG according to their priorities. These heuristics have two phases, namely the task priority phase and processor selection phase. In the first phase, the tasks are topologically sorted using rank or level attributes. And in the second phase, the best processor to reduce the overall completion time is selected. There are some states of art list scheduling heuristics Viz. Earliest Time First (ETF) [25], Levelized Min Time (LMT) [39], Dynamic Level Scheduling (DLS) [26], Critical Path On Processor (CPOP) [27], Modified Critical Path (MCP) [25], Heterogeneous Earliest Finish Time (HEFT) [27]. The DLS [26] algorithm dynamically changes priorities and schedules across both temporal and spatial dimensions to avoid shared resource contention. The merits of this algorithm are broadly targetable, high speed, flexible, and display better performance. LMT [39] works in two phases. The initial phase sets the tasks which are executed in parallel mode level-wise. The second phase is used to allocate the task to the best-fitted resource. HEFT [27] is used for a bound number of processors to the heterogeneous distributed systems having two phases, task priority and processor selection phases. In the first phase, tasks are sorted by descending/ascending order using upward/downward rank. The second phase selects the best resource to minimize the actual finish time of the tasks in the list. CPOP [27] uses a critical path in the given DAG. All tasks on the critical path are assigned to the single best processor, and the remaining tasks are allocated with the rank-based allocation same as HEFT. The authors of [28], proposed a list-based workflow scheduling method inspired by HEFT for non-preemptive periodic tasks, reporting better performance. DBEFT has been proposed [29]. DBEFT is an extended work of HEFT with the same objective. Security Aware DAG scheduling using ranking-based ordering and earliest finish time has been proposed, and superior performance has been reported in the domain [30]. In another work, level-based task ordering is used for security-oriented workflow scheduling, minimizing total number of failures [31]. Multi-objective version of levelized workflow task execution has been presented in to optimize makespan and flow time [32]. Another deadline-aware multi-objective model is reported considering execution time and monetary cost [33]. A workflow task allocation model is also developed with energy and dependency constraints for the heterogeneous environment [38].

Clustering workflow heuristics are developed to minimize the transfer time between dependent tasks. Since the list-based heuristic does not consider communication cost seriously, consequently generating needless idle gaps on machines as in EFT [40] and HEFT [27]. A clustering procedure has been employed as a sequence of clustering refinements. Clustering-based approaches consist of mapping tasks to clusters and ordering tasks within the cluster [41], [42]. This approach aims to attain an overall minimum communication time share among tasks by merging them in the same clusters and assigning them on the same machines at the cost of the sacrifice of parallelism within the DAG. Thus, a trade-off between minimizing communication delay and maximizing parallelism has been observed [43]. There are some clustering-based algorithms such as Linear Cluster Method [44], Dominant Sequence Clustering [45], Resource Aware Clustering (RAC) [46], and Clustering for Minimizing the Worst Schedule Length (CMWSL) [47]. RAC [46] aims to achieve relative load balancing and efficiency improvement for a machine with different capacities. For this, RAC gives a dynamic score function for all tasks, followed by task clustering and then task allocation per the processors' computing capability. CMWSL [47] is a cluster-based task scheduling algorithm with four phases to minimize the scheduling length for more heterogeneous processors.

On the other hand, in multiple workflow allocations, all scheduling decisions are made after a successive activation period. An estimated number of workflows with similar requirements have arrived and form batch results in an efficient schedule. The work proposed in [13] presents aggregated DAGs-based multiple workflow allocation models allocating tasks according to depth level to optimize throughput. In this work, tasks within the workflow are non-communicating, and inter-task communications among them have been considered zero. Author of [48] presents four approaches, i.e., sequential (one after another), gap search (next DAG utilizes gaps), interleaving and grouping of DAGs (merging DAGs into a single one with one entry and exit node). These approaches evaluate each other by using a modified Path Clustering Heuristic (PCH) for prioritization of tasks and selection of machines. Other work [49] proposed two workflow allocation strategies, viz. MWGS4 (Multiple Workflow Grid Scheduling 4 stages) and MWGS2 (Multiple Workflow Grid Scheduling 2 stages). These strategies have various stages Viz. Labeling, adaptive allocation, prioritization, and parallel machine scheduling. In [22], the authors solve the problem of multiple workflow scheduling with four policies for DAGs allocation and another two focusing on fairness optimizing makes pan with good fairness has been presented. The paper [50] developed a RANK hierarchical considering communication contention with task dependencies (CCRH) for reliable scientific workflow allocation in the cloud. CCRH aims to maximize reliability and improve system fault tolerance. In [51], the authors introduced a novel dynamic task rearrangement and rescheduling approach to allocate multiple workflows considering resource efficiency and robustness. The rearrangement policy improves robustness. In [52], the authors proposed a cluster-based approach for multiple workflows with soft deadlines to examine the effect of time restriction on the quality of task allocation in a heterogeneous environment. In this paper, the authors consider how to accommodate free time windows by tasks from workflows to fulfill the user's requirements on an urgent basis. In [53] and [54], the authors have proposed an energy-aware stochastic method for multiple DAGs on heterogeneous DVFS-enabled machines to minimize the energy and time. A model for the batch of DAGs with precedence constraints maximizing load balancing level for effective resource usage has been presented in [55]. A level-based batch scheduling strategy (SLBBS) [23] minimizing the turnaround time with inter-task communication has been presented in the literature. Also, its performance evaluation has been carried out with some other methods in [56]. In [57], the authors have proposed a multi-objective workflow model for multi-DAGs optimizing the completion time and monetary cost.

III. THE PROPOSED MODEL

This Section describes the proposed Levelized Multiple Workflow Scheduling Strategy with Task Merging (LMWS-TM) to optimize the turnaround time of the multiple workflows submitted for processing on VMs for IaaS cloud computing. Moreover, this Section presents various things, e.g., the list of notations used, cloud system framework, VM model, workflow applications model, problem statement, parameter estimation, the algorithm used with illustrations, motivations of the work, and time complexity of the algorithm.

A. CLOUD SYSTEM FRAMEWORK

This Section introduces the Infrastructure as Service cloud system architecture for workflow allocation, as shown in Figure 1. The various components of the architecture have been explained as follows:

1) CLOUD USERS

Multiple cloud users have their workflow applications. They need to submit their requests/applications over the cloud system for processing on some negotiated rent. Users aim to fulfill their requirements by satisfying constraints to optimize QoS parameters.

2) WORKFLOW APPLICATIONS

As shown in Figure 1, a batch of multiple workflows is submitted from many users at different locations and grouped at the arrival queue on the central dispatcher for further processing. Each workflow comprises many dependent and cooperative tasks with precedence constraints having parent and child relationships. Workflows are independent of one another and represented by using a Direct Acyclic Graph (DAG), as depicted in Figure 2. The detailed description and

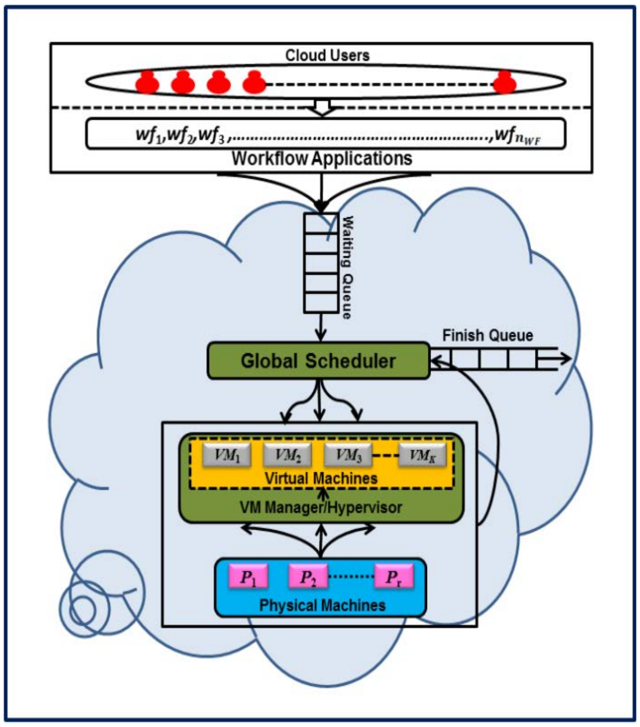

FIGURE 1. Cloud system framework.

modeling of the batch of multiple workflows are presented in section III C.

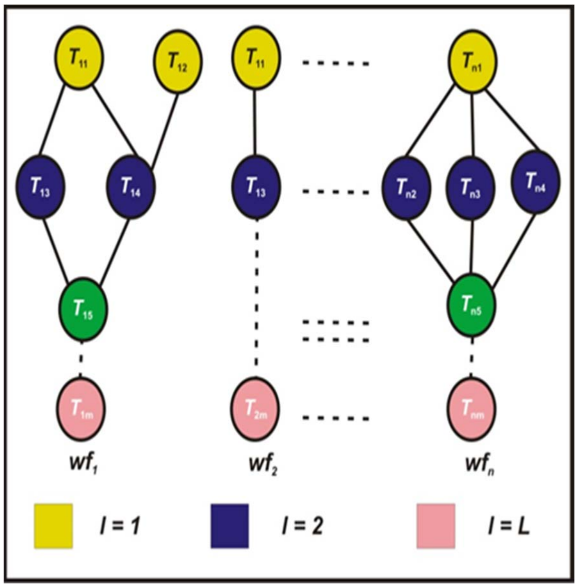

FIGURE 2. A sample multiple workflows.

3) WAITING QUEUE

A waiting queue is maintained at the global scheduler to accommodate incoming workflow applications there from different users to form a batch. These workflows are then dispatched for processing by a global scheduler on compute-intensive VMs in the cloud. The waiting time of applications in this queue till mapping is known as queuing time. We consider this queue follows an M/M/1 model. This model has infinite queue length with a single server. Also, workflow arrival and service times are determined by a Poisson and exponential distribution, respectively. The arrival rate and service rates are λ and μ respectively. The queue length is assumed infinite in this case.

4) GLOBAL SCHEDULER

Global Scheduler is one of the essential components in this framework. It is responsible for capturing all required information about the waiting queue, workflow applications, virtual machines, and associated physical machines. The global scheduler performs workflow allocation effectively by using data captured. The main aim of the global scheduler is to distribute the workflow components over virtual machines created by the hypervisor to improve system performance. It plays a crucial role in cloud systems. However, static centralized workflow allocation has the benefit of generating an efficient schedule for a batch of workflows onto a set of VMs due to having prior estimated knowledge of various application characteristics and resources.

5) PHYSICAL MACHINES

The physical machines comprise clusters, supercomputers, servers, etc., the fundamental physical computing resources that make up a cloud infrastructure. Through virtualization, users can use the virtualized versions of machines of the physical machine without any management overhead.

6) VM MANAGER/HYPERVISOR

The hypervisor can be viewed as a software layer that provides flexible management of virtual resources to users in an application layer. This cannot be used directly by the end-users. A connection is needed between any two entities like clients, servers, or applications. VM monitor enables virtual operating systems (OS) to run simultaneously on a machine. The hypervisor provides the number of VMs, computing capacity, bandwidth, storage, etc. The VM manager decides on allocating tasks, reducing resource cost, time, etc.

7) VIRTUAL MACHINE

A Virtual Machine is an application environment installed on software that mimics the behavior of a dedicated physical machine. The cloud service provider provides VM. The users have the same experience as they would have on the physical machines. A detailed VM modeling has been presented in coming to section III B.

8) FINISH QUEUE

The workflow tasks submitted by users are pooled in the finish queue after their completion. The finish queue acts as a buffer of results from processing. As depicted in Figure 1, generally, the relation among the number of physical machines, VMs, and workflows is $r \ll k <= n_{WF}$.

B. VM MODEL

Now, CSP offers a set of K virtual machines VM = {VM_1, VM_2, ….,VM_k}. The following characteristics regarding VMs from the IaaS CSP are listed as:

- K number of virtual machines (VMs) for mapping multiple independent workflows.
- Machines are capable of compute-intensive workflows.
- Computing capacities of VMs (CC_k)
- Initial Ready time (RT_k) which measures the previous load on VM_k.
- VM distances (D_{ab}) are the distance between VM_a and VM_b estimated as the number of links.
- A matrix, E ($n_{WF} \times n_{wf_i} \times$ K), written as E_{ijk} is the Expected Time to Compute of T_{ij} in the workflow wf_i on VM_k.

C. WORKFLOW APPLICATIONS MODEL

A batch of multiple workflows WF = {wf_i: $1 \leq i \leq n_{WF}$} has been considered in which each workflow is modeled by using a Direct Acyclic Graph (DAG). Each workflow comprises of a set of tasks $\tau_i = \{T_{ij} | 1 \leq i \leq n_{WF}, 1 \leq j \leq n_{wf_i}\}$ and set of links (edges) between the tasks in the workflow. In the batch of workflow, each workflow consists of various tasks having depth levels and may need inter-task communication with predecessor tasks from the same workflow at previous depth levels. The following characteristics of workflows are listed as

- A batch of n_{WF} the number of compute-intensive multiple workflows represented by DAG.
- Each task T_{ij} in workflow wf_i is associated with a unique level attribute l_{ij}.
- Level attributes have been used to manage precedence and dependence constraints.
- L_i is the depth level of workflow wf_i(L_i = max(l_{ij}: precedence level of all $T_{ij} \in wf_i$)).
- Depth level of the whole batch of the workflow (L) is the maximum of L_i, i.e., L= max(L_i).
- Inter-task communication ($Data_{ixy}$) is between tasks T_{ix} and T_{iy} has been considered and measured in MIs.
- Multiple workflows have been divided into L partitions (ρ^L) as per depth levels.

A batch of multiple workflows is presented in Figure 2, where tasks in the workflows are sliced into groups as per the order of precedence. They may require communication with the other tasks at previous levels in the workflow. Workflows in WF may have different depth levels. The same color is used to represent the tasks at the same depth level in the workflows, e.g., T_{11}, T_{12}, T_{21} …T_{N1} has precedence level 1 while T_{13}, T_{14}, T_{22}…T_{N4} has level 2 and is shown by yellow and light blue, respectively. Moreover, it is evident from Figure 2 that T_{15} is dependent on T_{13} and T_{14} to begin its execution. Tasks at the same depth level e.g., T_{13}, T_{14}, T_{22}…T_{N4} can be processed simultaneously. Finally, it is supposed that the workflow pre-processing has been done before mapping the workflows.

Communication cost ($CC_{ixyab}^{l-s,l}$) in the considered scenario depends on inter-task communication ($Data_{ixy}$) and machine distance (D_{ab}) between machines VM_a and VM_b. $CC_{ixyab}^{l-s,l}$ between two tasks $T_{iy} \in \rho^l$ and $T_{ix} \in \rho^{l-s}$ of workflow wf_i (T_{iy} is assumed to be dependent on T_{ix}) allocated on machine VM_a and VM_b, respectively, as presented in Figure 3 and can be estimated as follows [23]:

$$CC_{ixyab}^{l-s,l} = z\left(Data_{ixy} \times D_{ab}\right) \tag{1}$$

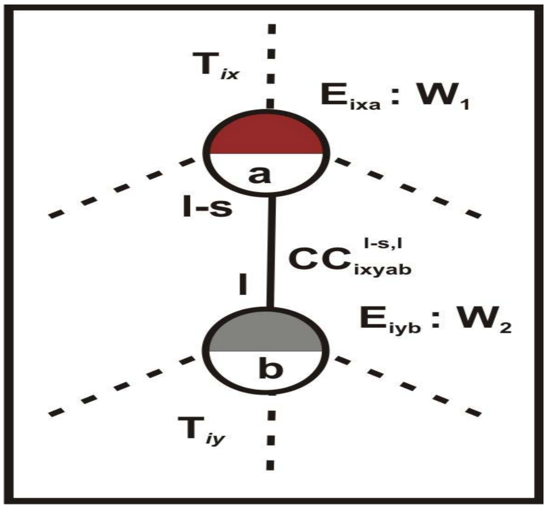

FIGURE 3. Communication cost computation.

Here, s∈ z + s ≥ and x, y=1, 2, 3…n_{wf_i}. The communication cost ($CC_{ixyab}^{l-s,l}$) is direcly proportional to both $Data_{ixy}$ and D_{ab}. The z is the constant of proportionality with linear relationship between them.

Therefore, the value of z is considered unity.

D. PROBLEM FORMULATION

The workflow allocation problem is the mapping Ø for the set WF = {wf_i: $1 \leq i \leq n_{WF}$} submitted for execution on the set of virtual machines (VMs) in an IaaS cloud environment to produce an allocation schedule (AS) with the aim of optimizing the objective criteria:

$$\emptyset : WF \rightarrow VM \tag{2}$$

Here, the objective is the turnaround time of the submitted set of workflows (WF) subject to the constraints as

1. $\sum_{j=1}^{n_{wf_i}} Allocation\ [i][j][k] = P; \quad 0 \leq P \leq n_{wf_i}$
2. $\sum_{k=1}^{K} Allocation\ [i][j][k] = 1$
3. $\sum_{i=1}^{N}\sum_{j=1}^{n_{wf_i}}\sum_{k=1}^{K} Allocation\ [i][j][k] = \sum_{i=1}^{n_{WF}} n_{wf_i}$
4. $EST(T_{ijk}) \geq max\left\{AFT\left[pred\left(T_{ij}\right)\right]\right\}$

i.e., the allocation must satisfy precedence constraints for all T_{ij}.

Allocation starts with merging tasks of the workflows following the procedure as explained in detail in Section IIIC. Afterward, the allocation of the tasks of each depth level has been done based on the allocation method mentioned, and detailed illustrations have been presented for the same in sections III C. Level-wise allocation and execution of the workflows having L depth levels with their execution time, communication time, and ready time values for various depth levels have been presented by using different colors in Figure 4.

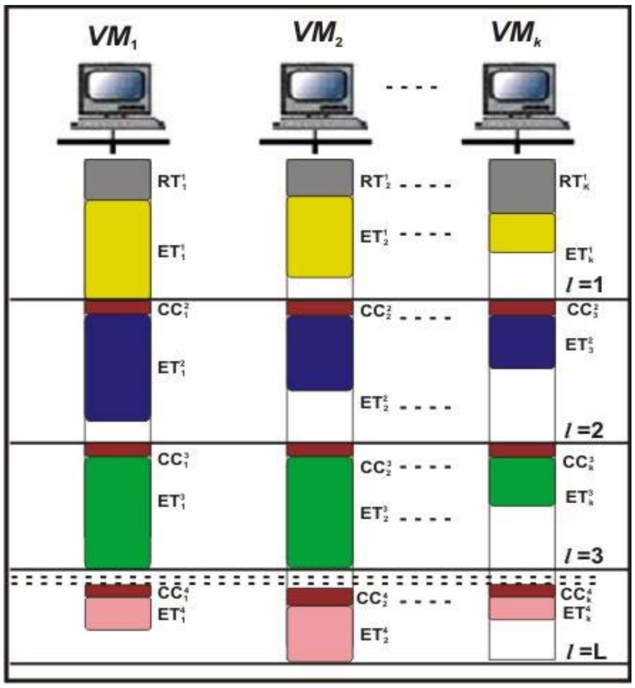

FIGURE 4. A sample level-wise execution of workflows.

Execution time at level l on VM_k is the sum of E_{ijk} of assigned tasks on the specified virtual machine and can be calculated as

$$ET_k^l = \sum_{VM_k \leftarrow \forall T_{ij} \in \rho^l} E_{ijk} \times \text{Allocation } [i][j][k] \tag{3}$$

E_{ijk} is an expected time to compute of T_{ij} and Allocation[i][j][k] is allocation vector having binary values and both can be written by equations as follows

$$E_{ijk} = \frac{size(T_{ij})}{CC_k} \tag{4}$$

$$Allocation\ [i]\,[j]\,[k] = \begin{cases} 1; & when\ VM_k \leftarrow T_{ij} \\ 0; & Otherwise \end{cases} \tag{5}$$

The net communication cost (CC_k^l) is taken as the maximum of the communication cost of the tasks assigned on VM_k with their predecessor tasks. Since the communication requirements of next-level tasks can be met in parallel after the execution of previous-level tasks, the CC_k^l of assigned tasks to their predecessor tasks on VM_k at depth level l can be estimated and written as

$$CC_k^l = \max_{VM_k \leftarrow \forall T_{ij} \& VM_h \leftarrow pred(T_{ij})} \left(CC_{ixyab}^{l-s,l} \right) \tag{6}$$

where $1 \leq h \leq k$, CT_k^l is the total completion time on VM_k and computed as the sum of RT_k, CC_k^l and ET_k^l as per the equation written as

$$CT_k^l = \begin{cases} RT_k^l + ET_k^l & if\ T_{ij} \in \rho^1 \\ CC_k^l + ET_k^l & otherwise \end{cases} \tag{7}$$

RT_k^l is the ready time of VM_k, which is the workload assigned prior to allocation and incorporated at the first depth level only. The turnaround time (TAT) is estimated as the time duration between workflows submission to completion and can be written as:

$$TAT = QT + \sum_{l=1}^{L} \max_{\forall k} \left(CT_k^l \right) \tag{8}$$

where QT [23] is the average queuing time of workflows in the central queue and can be expressed as

$$QT = \begin{cases} \frac{1}{\mu - \lambda}, & B \leq Q_u \\ \frac{1}{\mu - \lambda} \left(\frac{B}{Q_u} \right), & B \geq Q_u \end{cases} \tag{9}$$

where queue unit (Q_u) of servicing node is the amount, it can handle in one go, and B is the total number of MIs in the batch. Flow time can be computed as:

$$FT = \sum_{l=1}^{L} \sum_{k=1}^{K} CT_k^l \tag{10}$$

The average system utilization (U_S) of the resources in the cloud system can be estimated as:

$$U_S = \frac{\sum_{l=1}^{L} \sum_{k=1}^{K} CT_k^l}{K * \text{SUM}\left(\max_{\forall k} \left(CT_k^l \right) \right)} \tag{11}$$

Finally, the Response time (RT) is the time duration between the workflow's admission in the system and the appearance of the first responses of execution of the task from the system. It is a significant parameter to measure the interactive-ness of the system towards user's workflows. RT for the batch multiple workflows is calculated as

$$RT = \frac{1}{n_{WF}} \sum_{i=1}^{n_{WF}} \min_{T_{ij} \epsilon \rho^1 \& \forall wf_i} EST(T_{ijk}) \tag{12}$$

The proposed model always desires the optimum values of QoS parameters for the resultant allocation schedule.

E. LEVELIZED MULTIPLE WORKFLOW SCHEDULING STRATEGY WITH TASK MERGING

The method starts with partitioning the batch of multiple workflows per the level attribute in the DAG. And then, the scheduler first merges the tasks of workflows in batch in such a manner that overall execution time gets reduced. The task merging starts with determining the largest tasks as per the size across the workflows at each depth level, observed from first to the last. An effort is made to package the tasks of successive precedence levels for each workflow in such a way that the resultant size has a size less than or equal to the chosen largest task in the given depth level. The logic behind this packaging is that if there is a considerable size difference between the tasks selected for execution at the given depth level, the nodes will be idle till the largest task is over with its execution. The logic behind this task merging is that if there is a huge difference between the tasks selected for execution at the given depth level, the VM will be idle until the largest task is over with its execution. Therefore, the proposed scheme tries to fill the expected gaps produced due to the size difference between tasks at the same depth level. This, in turn, ensures a better total execution time and utilization with the benefits depending on the various attributes of the workflows comprising the batch viz. the degree of dependence, number of tasks at the given depth level, number of depth levels, and the available hardware parallelism in terms of the number of VM. Afterward, this batch of workflows gets assigned as accordance with the depth level on the appropriately selected VM as decided by the scheduler as per the allocation policy. This exercise's overall result is only for effectively exploiting the parallel portion and managing communication cost share in the workflows on VM while maintaining the order of execution of tasks in the workflows. It is also observed that the performance of this scheme is well suited for heterogeneous VM selected for computation are of the comparable capacities. The procedural steps for the same are presented as follows: The procedure for the task merging is given from step 4 to step 14 as in Algorithmic Template. Further, the regrouped tasks get allocated level-wise from the workflows on the appropriately selected VM. The allocation pattern of the tasks of the batch of workflows has been inherited from the allocation policy given in our previously proposed work [23]. However, the allocation pattern is modified with the expectation of enhancement in the turnaround time. Here, at each time largest or smallest tasks from the specified partition get selected for allocation. After that, the scheduler searches for a suitable virtual machine to allocate the tasks belonging to the partition at hand. The algorithm for the allocation of tasks of the batch of DAGs as per their depth level is given in the template from step 15 to step 31. With the amalgamation of task merging (TM), the work intends further to improve the combined scheme's performance, namely LMWS-TM. The scheme is capable enough to be adopted by any other scheme, which is precedence based on fine-tuning the tasks assigned to each level. Finally, for each partition, the values of CC_k^l and CT_k^l are computed. The algorithmic template for the same has been presented as follows:

LMWS-TM Input: *WF, T_{ij}, VM_k, E_{ijk}, $Data_{ixy}$, D_{ab}, QT, size(T_{ij}), n_{WF}, n_{wf_i} and K*
Output: *Allocation schedule (i, j), TAT, U_s, FT, and RT*

```
1. Divide WF as per depth level in $\rho^l$ // Divide the batch as per
   depth levels
2. Compute $E_{ijk}$ as per equation (4)
3. Sorting all $\rho^l$ // Sorting partition ascending/ descending order
4.  {
5.    for all $\rho^l$ do
6.              Get $size^l_{max}$ and $size^l_{min}$
       // first and last task from the sorted $\rho^l$
7.                Select $T_{ij}$ i.e. size($T_{ij}$)=$size^l_{min}$
8.                    do until $size^l_{min} \leq size^l_{max}$
9.                         $T_{ij} \leftarrow T_{ij} \oplus succ(T_{ij})$
10.                        $size^l_{min} = size^l_{min} + size(succ(T_{ij}))$
11.                        $d\_level\ (T_{ij} \oplus succ(T_{ij})) \leftarrow_l$
12.                   end do
13.   end for
14. }
15. for l=1 to L do
16. for all $T_{ij} \in \rho^l$ do
17.        for all $M_k$
18.    if (l=1)
19.               $VM_k \leftarrow T_{ij}$ // with least $ET^l_k + E_{ijk} + RT^l_k$
20.               else
21.               $VM_k \leftarrow T_{ij}$ // with least $ET^l_k + E_{ijk}$
22.    end if
23.  end for
24.  Alloc(i, j) $\leftarrow$ k
25.       Update Execution Time $ET^l_k$ //$ET^l_k$= $ET^l_k$+$E_{ijk}$
26.   $\rho^l$=$\rho^l$-$T_{ij}$ // Remove allocated tasks from the partition
27. end for
28.  Compute $CC^l_k$ // as per equation (6)
29.  Compute $CT^l_k$ // as per equation (7)
30. end for
31. }
32. Find Allocation Schedule (i, j) and Calculate TAT, FT, $U_s$, and
    RT // as per equation (8) −− (12)
```

An illustration to explain LMWS-TM, SLBBS, HEFT-1 and HEFT-2 has been presented in this Section with two workflows consisting of 6 tasks with a precedence/depth level among themselves, as shown in Figure 5. Let the arrival rate of the workflows and service rate of the central dispatching virtual machine be 0.03 and 0.05-unit time, respectively. And queuing unit (Q_u) is 10,000 MIs.

We have taken only three VM, but there may be more in real scenarios. The complete information on heterogeneous VM is shown in Figure 6, for example, the computing capacity (CC_k) and initial ready time (RT_k) of V_1 are 9 and 40 respectively. And VM distance (D_{12}) between V_1 and V_2 is 2. In each workflow, information like the number of instructions, depth level, and inter-task distance between two tasks of the same workflow is presented in Figure 5. For example, the size of T_{11} is 1100 Mis and the inter-task communication ($Data_{113}$) between T_{11} and T_{13} is 3. E_{ijk} for tasks of each workflow can then be computed and shown in Table 1.

Now, partitions of the batch as per depth level are $P^1 = \{T_{11}, T_{12}, T_{21}, T_{31}\}$ $P^2 = \{T_{13}, T_{14}, T_{22}, T_{23}, T_{24}\}$, $P^3 = \{T_{15}, T_{16}, T_{25}\}$ and $P^4 = \{T_{26}\}$. In wf_2, the tasks T_{21} and T_{22} of depth level 1 and 2 are combined as per the

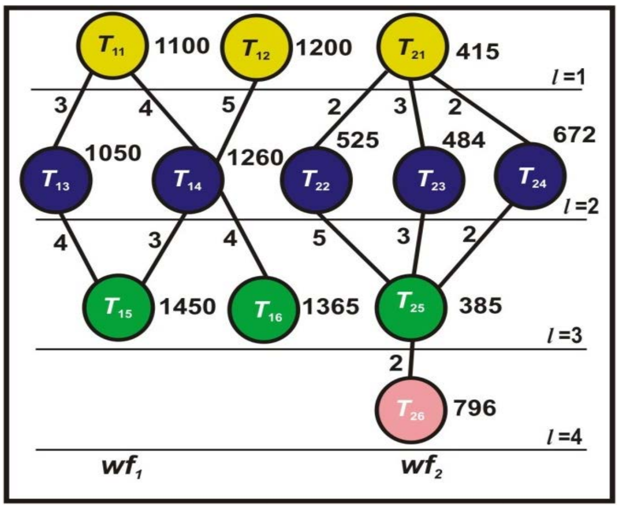

FIGURE 5. Sample of Two Workflows.

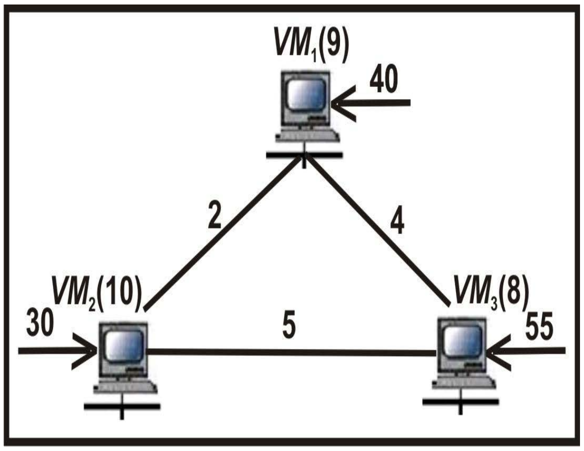

FIGURE 6. VM information.

TABLE 1. E_{ijk} of workflows for different VM.

WF		T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}
Wf_1	VM_1	122.22	133.33	116.67	140	161.11	151.67
	VM_2	110	120	105	126	145	136.5
	VM_3	137.5	150	131.25	157.5	181.25	170.625
		T_{21}	T_{22}	T_{23}	T_{24}	T_{25}	T_{26}
Wf_2	VM_1	46.11	58.33	53.78	74.67	42.78	88.44
	VM_2	41.50	52.50	48.40	67.20	38.50	79.60
	VM_3	51.875	65.625	60.50	84	48.125	99.50

task merging policy mentioned in the algorithmic template. Similarly, tasks T_{25} and T_{26} of depth levels 3 and 4 also are merged. First workflow wf_1 requires no merging of tasks. The resultant batch of workflow can be seen in Figure 7. The ETC matrix also requires updating as per the previous merging, as shown in Table 2.

The workflows are grouped into partitions according to precedence level, then sorted in descending order i.e.,

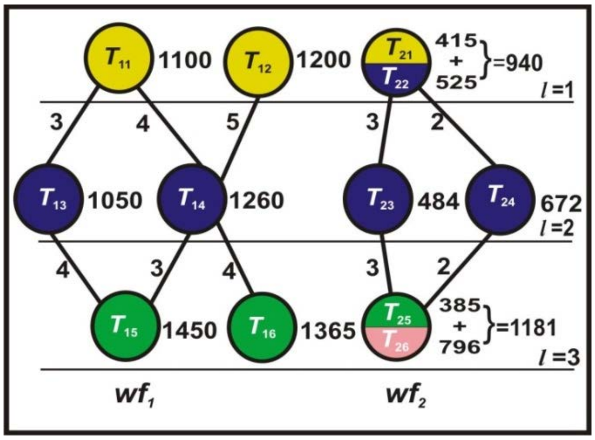

FIGURE 7. Workflows after task merging.

TABLE 2. E_{ijk} after task merging.

WF		T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}
Wf_1	VM_1	122.22	133.33	116.67	140	161.11	151.67
	VM_2	110	120	105	126	145	136.5
	VM_3	137.5	150	131.25	157.5	181.25	170.625
		T_{21}+ T_{22}		T_{23}	T_{24}	T_{25}+ T_{26}	
Wf_2	VM_1	103.87		53.78	74.67	131.22	
	VM_2	93.5		48.40	67.20	118.8	
	VM_3	117.50		60.50	84	147.625	

$L_1 = \{T_{12}, T_{11}, T_{21}\}$, $L_2 = \{T_{14}, T_{13}, T_{24}, T_{22}, T_{23}\}$, $L_3 = \{T_{15}, T_{16}, T_{25}\}$ and $L_4 = \{T_{26}\}$. Before allocation of tasks from the first partition, compare the execution time of T_{12} and T_{21} onto each virtual machine. The differences are 76.11, 68.5, and 85.625 on each virtual machine. So, check the execution time value of the succeeding task of T_{21}. Succeeding Task T_{22} execution time values are 58.33, 52.50, and 65.625 on VM_1, VM_2, and VM_3, respectively. Thus, tasks T_{21} and T_{22} can be merged and executed both simultaneously. In the second level, task merging cannot be possible due to T_{23} and T_{24} will be fulfilled. Similarly, in the third level, task merging is possible; tasks T_{25} and T_{26} are combined and executed both simultaneously same as the first level. After complete task merging, the depth level order as $L_1 = \{T_{12}, T_{11}, T_{21}+ T_{22}\}$, $L_2 = \{T_{14}, T_{13}, T_{24}, T_{23}\}$ and $L_3 = \{T_{15}, T_{16}, T_{25}+ T_{26}\}$. The merging of tasks level wised and combined value execution time value is depicted in Figure 7 and Table 2. All the execution of tasks is shown in Figure 8.

After task merging, the workflows are assigned on the selected VM as per the allocation strategy mentioned earlier, and the respective values of RT_k^l, CC_k^l and, ET_k^l as shown in Figure 8. Further, the performance parameters are computed as follows:

$CT_1 = 40+133.33+8+140+0+151.67 = 473$, $CT_2 = 30+110+0+105+6+145 = 396$

$CT_3 = 55+117.50+144.50+147.625 = 464.625$.

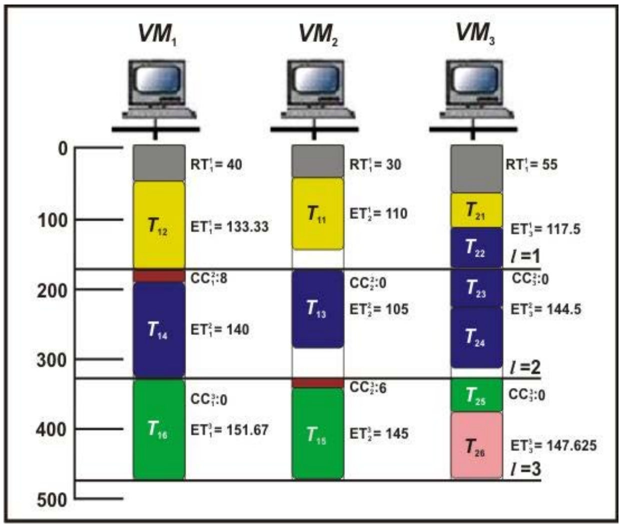

FIGURE 8. Allocation of tasks using LMWS-TM.

TAT= **Max (473,**396, 464.625) = 473, FT = 473+396 +464.625 = 1333.625

U_s = 1333.625/3*473= 0.9398, RT = (30+55)/2= 42.5

Using SLBBS, the allocation of the workflows has been done level-wise with tasks allocated on VM_k. and respective values of RT_k^1, CC_k^1 and, ET_k^1 as shown in the Figure 9. The detailed illustrations can be seen in [23]. Further, the performance parameters are computed as follows:

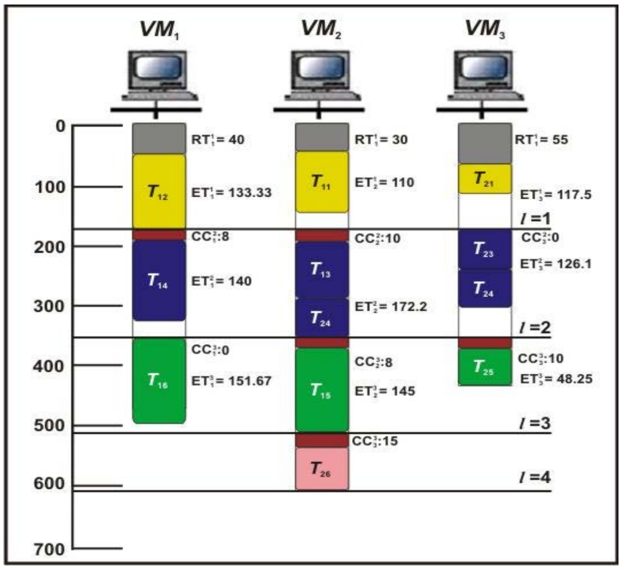

FIGURE 9. Allocation of workflows using SLBBS.

CT_1 = 40+133.33+8+140+151.67 = 473, CT_2 = 30+110+10+172.20+8+145+15+79.60 = 569.80

CT_3 = 55+51.875+125.625+10+48.125 = 290.625

TAT=Max (473, 569.80, 290.625) = 569.8, FT = 473+569.80+290.625 = 1333.425

U_s = 1333.425/3*569.80= 0.7801, RT = (30+55)/2= 42.5

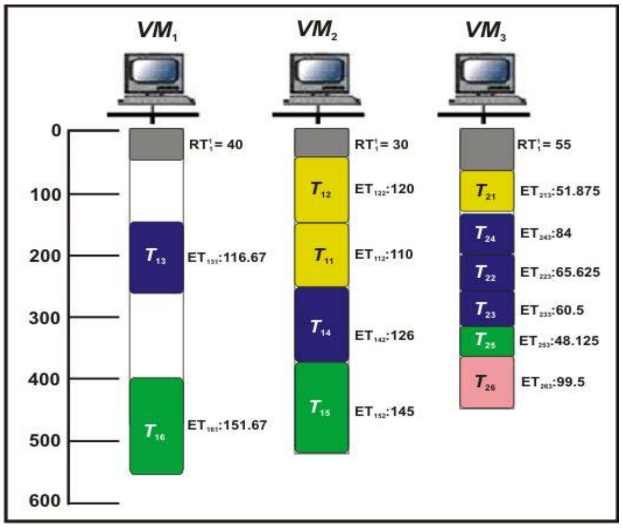

FIGURE 10. Allocation of tasks using HEFT-2.

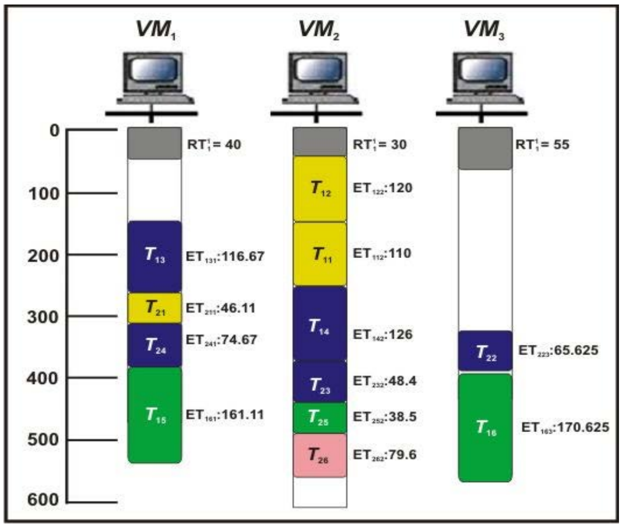

FIGURE 11. Allocation of tasks using HEFT-1.

The same workflows are allocated using HEFT [27], and performance parameters are computed. HEFT is implemented for multiple workflows such as HEFT-1 and HEFT-2 in two different approaches, viz. sequential and merged-based approaches are presented in [48] as shown in Figure 10 and Figure 11. In HEFT-1, workflows are allocated from batch one after another using HEFT as in [26]. On the other hand, HEFT-2, one pseudo entry, and one pseudo exit tasks form a single larger workflow from all, multiple workflows by joining initial and end tasks to pseudo entry and pseudo exit task, respectively. After this, HEFT is applied to the resultant DAG. Now, the values of parameters for HEFT-2 computed are as

CT_1 = 40+116.67+151.67 = 308.34, CT_2 = 30+110+120+126+145 = 531

CT_3 = 55+51.875+125.625+10+48.125 = 464.625

TAT=Max (308.34, 531, 464.625) = 531, FT = 308.34+531+464.625 = 1303.965

U_s = 1303.965/3*531= 0.7801, RT = (30+55)/2= 42.5

Similarly, the values of parameters for HEFT-1 for the same workflows computed are as

CT_1 = 40+116.67+46.11+74.67+161.11 = 438.56, CT_2 = 30+110+120+126+48.4+38.5+79.6 = 552.5, CT_3 = 55+65.625+170.625 = 291.25

TAT= Max (438.56, 552.5, 291.25) = 552.5, FT = 438.56+552.5+291.25 = 1282.31

U_s = 1282.31/3∗552.5= 0.7736, RT = (30+267.67)/2= 148.835

TABLE 3. Computed QoS parameters for the illustration.

Strategies	TAT	Us	FT	RT
LMWS-TM	473.00	0.9398	1333.625	42.50
HEFT-2	531.00	0.8185	1303.965	42.50
HEFT-1	552.34	0.7736	1282.310	148.83
SLBBS	569.80	0.7801	1333.425	42.50

The parameter values computed as per the illustration and shown in Table 3, LMWS-TM, HEFT-2, HEFT-1, and SLBBS strategies are represented along with the performance metrics mentioned in the above example. LMWS-TM is giving superior values 473 and 0.9398 on TAT and system utilization, while the average response time is almost the same as LMWS-TM and SLBBS. Also, HEFT-1 shows the best value among all on flow time. HEFT-1 has the worst value among all on average response time.

F. TIME COMPLEXITY

The time complexity is estimated for the proposed algorithm as per the steps involved in the algorithmic template in Section III E. Now, let the user has submitted n_{WF} number of workflows $WF = \{wf_i : 1 \leq i \leq n_{WF}\}$ and wf_i has n_{wf_i} the number of the tasks. The average number of tasks assumed here as n and t in the workflows and partitions, respectively i.e., $n = \frac{\sum_{i=1}^{i=n_{WF}} n_{wf_i}}{n_{WF}}$ and k number of VM. The depth levels for workflows (wf_i) are l_i, so, the depth level of WF is L= max (l_i: where i varies from 1 to n_{wf_i}). And, L can be estimated as L= log n. The time complexity of LMWS-TM involved partitioning the batch, sorting partitions, task merging, and allocation.

1) For dividing the workflows into partitions is $O(L \times n_{WF} \times n)$.
2) For sorting the partition is $O(L \times t \times t)$
3) For task merging is $O(L \times t \times t)$
4) For allocation is $O(L \times t \times M)$.

Thus, the overall time complexity is $O(L \times n_{WF} \times n) + O(L \times t \times t) + O(L \times t \times t) + O(L \times t \times M) \approx O(L \times t^2) = O(\log n \times t^2)$.

IV. EXPERIMENTAL STUDY

The experimental study has been conducted to observe and analyze the effect of task merging of workflow tasks in the batch of workflows on the set of VM available in the IaaS cloud system at the time of allocation. Statistical testing has also been conducted to check the significance level of the hypothesis derived from the study. The experiments were conducted using MATLAB 7.60 using Intel (R) Core (TM), i7-3770 CPU using 2GB RAM, and Sun Fire X4470 Server with 14 GB RAM. The study is done to evaluate the comparative behavior of the LMWS-TM with SLBBS, DLS, HEFT-2, and HEFT-1 for various cases to analyze its effectiveness in the middleware.

A. SIMULATION RESULTS

The simulation produces a realizing cloud environment and a batch of random workflows for evaluating the proposed strategy's performance. The multiple random workflows consist of parameters like workflow number (Batch Size), precedence level, task size, and degree of parallelism within the batch, amount of inter-task communication within workflow tasks. Parallelism in the batch of multiple workflows varies by varying the parallel tasks in the depth levels and vice versa. Similarly, IaaS cloud system has attributes such as number, computing capacities, distances of VMs. Accordingly, simulator prototype of the workflows allocator implemented in MATLAB produces associated input parameters of workflows, and VMs between a specific feasible limit are randomly generated by using a discrete uniform distribution. For the experimental results, the common parameter setting is given in Table 4 for all cases.

TABLE 4. The input parameters for experiments.

S. No.	Input Parameters	Range
1.	Number of workflows (n_{WF})	4-256
2.	Inter-task communication ($Data_{ixy}$)	1-1000
3.	VM distances (D_{ab})	1-100
4.	Number of virtual machines Initial (K)	8-128
5.	Computing capacities of VMs (CC_k)	30-1000
6.	Ready time (RT_k)	0-1000
7.	Number of depth level	4-64
8.	Arrival rate (λ)	0.03
9.	Service rates (μ)	0.05
10.	Queuing Unit (Qu)	20000

Experiments have been done by varying the input parameters related to the cloud system and batch of workflows for all considered strategies, i.e., batch size, number of VMs, and parallelism (depth level). The simulation results for each case are presented in figures and tables in this Section. All experiments are repeated by 50 times, and the mean of the corresponding parameters is reported for all the cases for avoiding the effect of randomness. For example, in varying batch size (for case 1), the experiments are conducted 50 times for each batch size, such as 4 to 128 and the mean values of turnaround time, utilization, response time, and flow time is reported in figure 12 to figure 15 and Table 5. The same pattern is used for other cases. Further, the response time of HEFT-1 is very large in comparison to LMWS-TM,

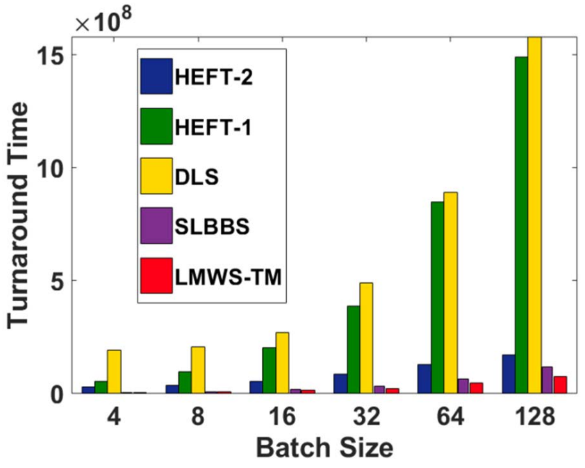

FIGURE 12. Turnaround time v/s batch size.

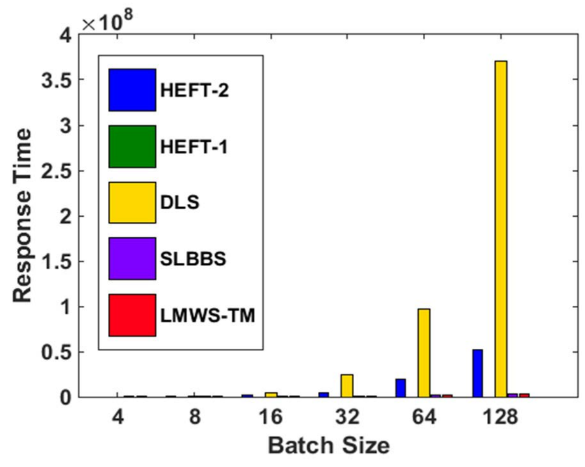

FIGURE 15. Response time v/s batch size.

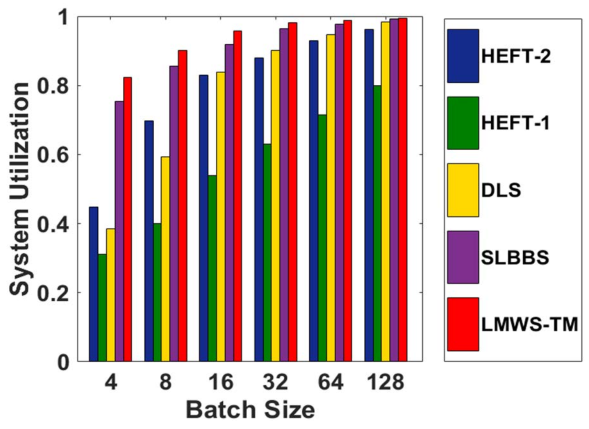

FIGURE 13. System utilization v/s batch size.

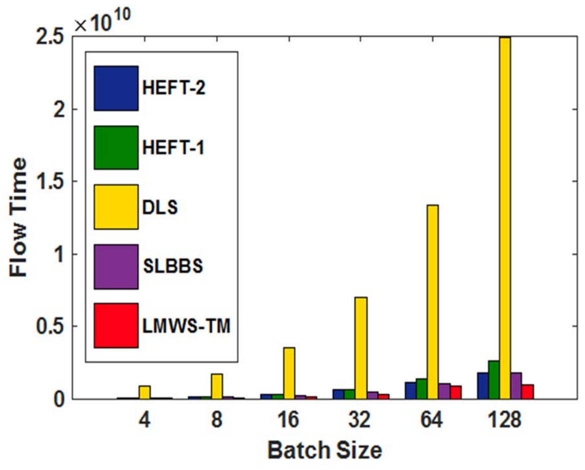

FIGURE 14. Flow time v/s batch size.

TABLE 5. QoS parameters of all methods for varying batch size.

n_{WF}	LMWS-TM	SLBBS	HEFT-1	HEFT-2	DLS
			Turnaround Time		
4	**5878834**	6998612	54132829	31784468	1.93E+08
8	**8498437**	10363948	96602445	35458823	2.09E+08
16	**14636065**	18526665	2.03E+08	54181278	2.72E+08
32	**24726553**	32968738	3.89E+08	85587214	4.91E+08
64	**4.75E+07**	6.69E+07	8.47E+08	1.31E+08	8.91E+08
128	**7.58E+07**	1.17E+08	1.49E+09	1.73E+08	1.58E+09
			System Utilization		
4	**0.82345**	0.75318	0.31191	0.44873	0.38416
8	**0.90213**	0.85637	0.40055	0.69808	0.59286
16	**0.95875**	0.91876	0.53981	0.82873	0.83809
32	**0.98187**	0.96327	0.63093	0.87865	0.90129
64	**0.98876**	0.97722	0.71380	0.92914	0.94752
128	**0.99396**	0.99288	0.80035	0.96124	0.98300
			Flow Time		
4	**66765432**	76330456	82340452	90677887	8.88E+08
8	**1.12E+08**	1.30E+08	1.56E+08	1.73E+08	1.73E+09
16	**1.99E+08**	2.61E+08	3.55E+08	3.49E+08	3.56E+09
32	**3.67E+08**	4.95E+08	7.01E+08	6.24E+08	7.01E+09
64	**8.99E+08**	1.04E+09	1.44E+09	1.15E+09	1.34E+10
128	**9.90E+08**	1.83E+09	2.68E+09	1.83E+09	2.49E+10
			Response Time		
4	40025	40155	2379605	4506	**73**
8	139034	138134	16437070	**104792**	212006
16	340211	**340201**	83758012	1292915	4326628
32	**778918**	783818	3.42E+08	4702733	24141591
64	**1669638**	**1675638**	1.70E+09	1.98E+07	9.72E+07
128	**3115393**	3123493	6.28E+09	5.17E+07	3.71E+08

SLBBS, HEFT-2. Therefore, for better presentation, HEFT-1 is omitted in figure 15, figure 19, and figure 23, and only four methods are presented in these figures. However, the numerical results are also presented in the tables for better clarity.

1) VARYING BATCH SIZE

This case focuses on the observations of the effect of varying batch sizes from $n_{WF} = 4$ to $n_{WF} = 128$, and the remaining input parameters are fixed as:

$k = 16, RT_{kl} = 0 - 1000, \mathcal{CC}_k = 30 - 1000, T_{ij} = 256$, $Data_{ixy} = 1 - 1000$, $L = 16, D_{ab} = 1 - 100$, $\mu = 0.05$, $\lambda = 0.03$, $Qu = 200000$

For better visibility in comparative analysis, the average values of TAT, Us, FT and RT have been also presented in Table 5 for varying workflows and the best values are shown in Bold.

Observations:

1) As shown in Figure 12, the trend of turnaround time keeps increasing as the number of workflows increase for all the strategies, viz. LMWS-TM, SLBBS, HEFT-1, HEFT-2, and DLS as expected on keeping all other input parameters fixed. As can be seen that the rate of increase in turnaround time by increasing the number of workflows is the least for LMWS-TM. LMWS-TM outperforms all other strategies for every batch size from small to larger. The performance order for turnaround time is LMWS-TM (best), SLBBS, HEFT-2, HEFT-1, and DLS (worst). The observed performance gain of LMWS-TM over SLBBS is in the range of 16% - 50% for 4 to 128 workflows in the batch. This performance gain is due to the amalgamation of task merging prior to allocation, which reduces the communication cost-share and the total number of depth levels resulting in improved TAT.
2) The trend of system utilization is increasing batch size and keeping other parameters fixed, as shown in figure 13. The performance order for utilization is LMWS-TM (best), SLBBS, DLS, HEFT-2, and HEFT-1 (worst). LMWS-TM outperforms all other strategies for every batch size in terms of system utilization. This is because as more workflows accumulate, more tasks become available at any precedence level, resulting in the batch exhibiting more parallelism within itself. LMWS-TM is designed keeping in mind to exploit parallelism at the workflow level as well as the task level with uniform allocation on the VMs available. Further, the proposed scheme tries to fill the expected gaps produced due to the size difference between tasks at the same depth level by task merging, ensuring better utilization. Also, the performance of LMWS-TM has been improved for batch sizes over SLBBS, which is significant for smaller batch sizes and tries to be at par for larger batches.
3) The trend of flow time increases as soon as batch size is raised for this case, as shown in figure 14. LMWS-TM outperforms all other strategies for batch sizes 4 to 128 in terms of flow time. LMWS-TM has improved in the range of 12.5% - 45.92% for batch sizes 4 to 128 SLBBS. The performance order is the same as the in the case of turnaround time with the reasons explained earlier.
4) Response time has an increasing trend when the batch size is increased for this case, as shown in figure 15. LMWS-TM and SLBBS both perform at pat for all batch sizes considered in the experiments on account of response time, and both are far better than other strategies. Response time is the average start time of the workflows, and due to level-wise allocation, every workflow execution starts at the first level at the beginning of execution for SLBBS and LMWS-TM. Task merging does not affect the first-level tasks. Hence, it has no effect on response time because it combines the tasks from the next depth levels and merges them with the tasks at previous levels. HEFT-2 performs better than DLS. HEFT-1 observes the worst performance because workflows are assigned one after another in serial exploiting only task level parallelism in the DAG.

2) VARYING DEPTH LEVELS WITH FIXED BATCH

This case observes the effect of variation in the number of depth levels in the fixed workflows because the degree of parallelism is changed at the depth level. Experiments are conducted here for $L = 4$ to $L = 128$, and the remaining input parameters are as follows:

$n_{WF} = 256, k = 32, RT_k = 0 - 1000, \mathcal{CC}_k = 30 - 1000, T_{ij} = 128, Data_{ixy} = 1 - 3000, D_{ab} = 1 - 100, \mu = 0.05, \lambda = 0.03, Qu = 200000$

Again, for better visibility in comparative analysis, the average values of TAT, Us, FT and RT have been also presented in Table 6 for depth levels and the best values are shown in Bold.

TABLE 6. QoS parameters of all methods for varying depth levels.

L	LMWS-TM	SLBBS	HEFT-1	HEFT-2	DLS
			Turnaround Time		
8	**21663078**	31395766	601254114	160953417	7.95E+08
16	**23892036**	32728817	630728134	199284558	8.04E+08
32	**27449840**	36599787	618078540	190768362	8.27E+08
64	**34595275**	44352918	627626691	130198744	7.95E+08
			System Utilization		
8	**0.9899876**	0.9888761	0.74934477	0.8575102	0.9871445
16	**0.9872110**	0.9851108	0.69538172	0.8348660	0.9836125
32	**0.9737521**	0.9637521	0.67866197	0.8262518	0.9650801
64	**0.9573492**	0.9473493	0.66733800	0.8170216	0.9520018
			Flow Time		
8	**877654321**	971269882	4307853842	2.322E+09	2.5E+10
16	**910865431**	991184236	4437252931	2.906 E+09	2.52E+10
32	**949865431**	1.065E+09	4628160030	3.280 E+09	2.56E+10
64	**981234514**	1.202E+09	4882542252	3.717 E+09	2.7E+10
			Response Time		
8	**1649286**	1651286	2184900939	57791402	2.29E+08
16	**697124**	696974	2274065457	38294582	1.46 E+08
32	271234	**264356**	2375991100	11405555	68610881
64	**70989.11**	71779.28	2331148868	642597.52	18830869

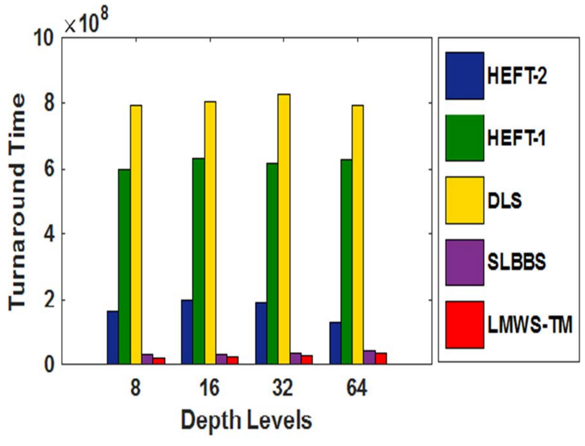

FIGURE 16. Turnaround time v/s depth level.

Observations:

1) As presented in Figure 16, turnaround time is gradually increased by increasing the depth levels for a fixed batch of 256 workflows. Increasing the number of levels reduces the parallelism in the workflows resulting in degrading turnaround time. LMWS-TM still performs best, and SLBBS is the second-best on turnaround time, followed by HEFT-1 and HEFT-2, with DLS worst. Performance gain of LMWS-TM over SLBBS for depth levels 8 to 64 is almost 22%-35%, respectively.
2) Average utilization is also slightly decreased for all the methods considered in experiments on varying the depth levels, as shown in Figure 17. Here, the batch size is 256. Therefore, LMWS-TM, SLBBS, and DLS are approximately at par. HEFT-1's performance is worst.

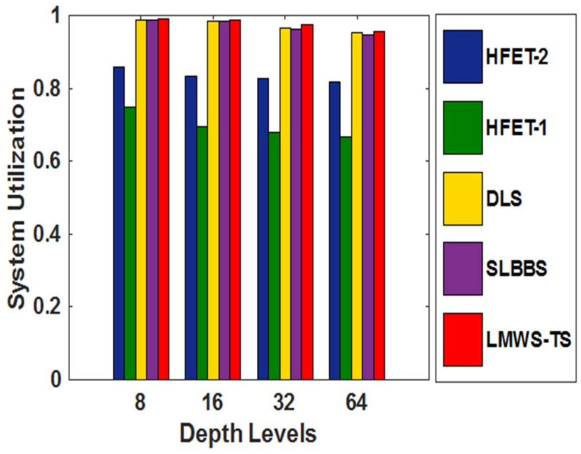

FIGURE 17. System utilization v/s depth level.

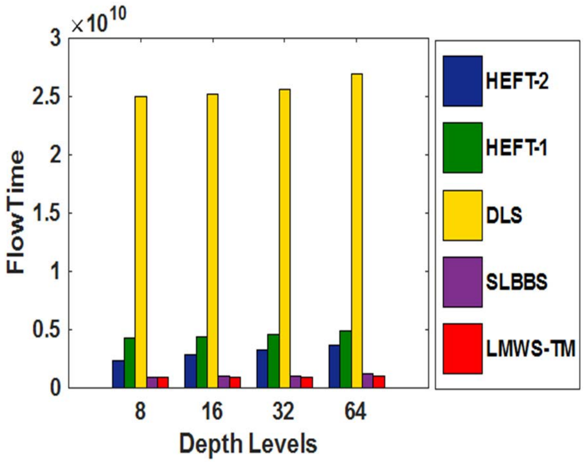

FIGURE 18. Flow time v/s depth level.

3) In this scenario, the flow time is gradually increased, as shown in Figure 18 for all the methods considered. The performance gain over SLBBS is almost 9%-19%.
4) In the case of varying depth levels (parallel tasks), as shown in Figure 19, LMWS-TM and SLLBBS are at par and significantly perform better than other peers.

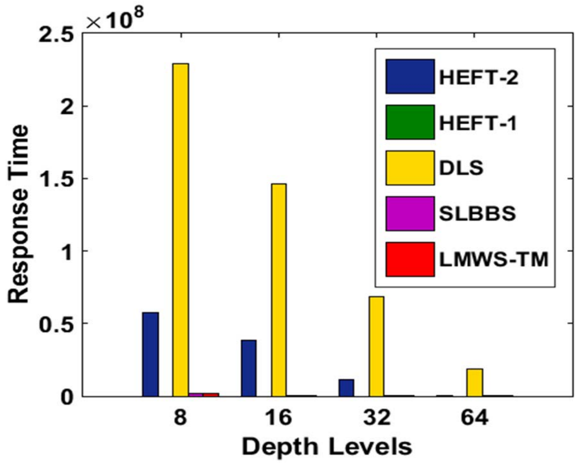

FIGURE 19. Response time v/s depth level.

Again, HEFT-1 performs worst for the same reason as mentioned earlier case in detail.

3) VARYING NUMBER OF VM

This case presents the effect of the variation in hardware parallelism (number of VMs) from k = 8 to k = 128, with the remaining input parameters fixed as follows:

$n_{WF} = 64, K = 32, RT_{kl} = 0 - 1000, \mathcal{CC}_k = 30 - 1000, T_{ij} = 512, Data_{ixy} = 1 - 100, L = 16, D_{ab} = 1 - 100, \mu = 0.05, \lambda = 0.03, Qu = 200000$

Again, for better visibility in comparative analysis, the average values of TAT, Us, FT and RT have been also presented in Table 7 for varying VMs and the best values are shown in Bold.

TABLE 7. QoS parameters of all methods for varying number of VMs.

K	LMWS-TM	SLBBS	HEFT-1	HEFT-2	DLS
			Turnaround Time		
8	**7.78E+08**	973090823	1.84E+09	1063196723	2.22E+09
32	**1.10E+08**	129724658	3.1E+08	158261931	7.51E+08
64	**45876512**	54681465	1.47E+08	81422682.8	8.01E+08
128	**20901234**	23993951	73641169	62391470	8.13E+08
			System Utilization		
8	**0.995567**	0.99467167	0.806733	0.9613011	0.844548
32	**0.96543211**	0.9592963	0.73859	0.94393147	0.720265
64	**0.92908765**	0.91210029	0.61849	0.91529898	0.629407
128	**0.7612345**	0.75881439	0.5727	0.6392402	0.567311
			Flow Time		
8	**6.96E+09**	7.72E+09	8.19E+09	8063193607	1.5E+10
32	**3.06E+09**	3.20 E+09	4.76E+09	4526804345	2.23E+10
64	**2.92E+09**	3.16 E+09	4.78E+09	4558457465	2.25E+10
128	**2.83E+09**	3.12 E+09	4.76E+09	4526804345	2.23E+10
			Response Time		
8	**28695652**	28728952	**6.89E+09**	302867984	7.78E+08
32	3770121	**3768451**	2.12E+08	7355491	39116973
64	1554126	1554126	28917560	**867543**	5860162
128	639263	639363	278574	**24141**	43111

Observations:

1) As presented in Figure 20, turnaround time has decreasing trend on increasing number of VM for LMWS-TM, SLBBS, HEFT-2, HEFT-1, and DLS as expected. LMWS-TM performs best for all

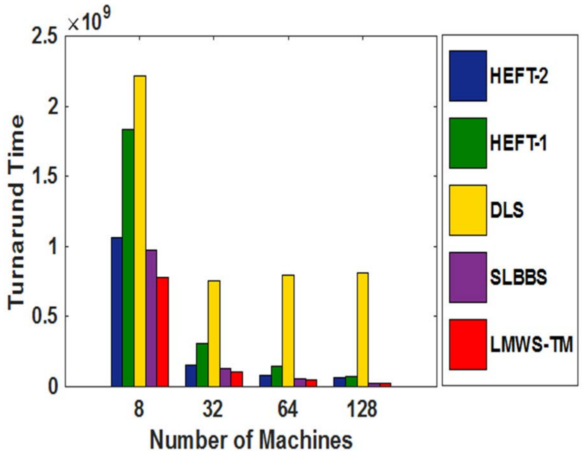

FIGURE 20. Turnaround time v/s number of VM.

configurations of VM among all strategies taken in experiments. The performance order is the same as in the first case i.e., LMWS-TM, SLBBS, HEFT-2, HEFT-1, and DLS. The reason for the superior performance of LMWS-TM has been explained earlier in detail. The improvement in turnaround time for the case is reported as almost 15%-30% for the 2 to 128 VM. Chances of selecting better machines increase with a larger number of VM. Consequently, the performance of the strategies is improved as VMs instances are increased.

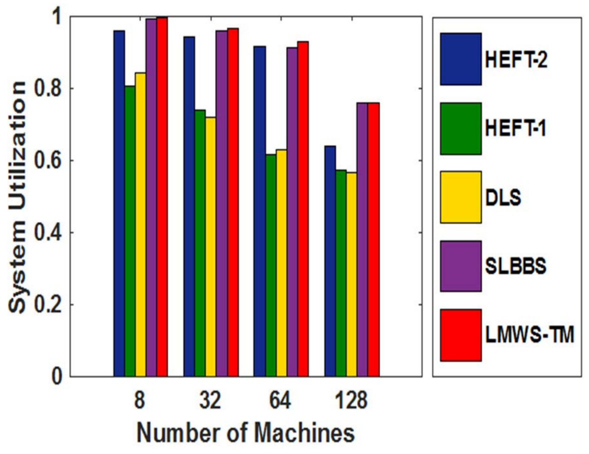

FIGURE 21. System utilization v/s number of VMs.

2) As in Figure 21, average utilization also shows a decreasing trend in increasing the number of VM. The performance order is LMWS-TM, SLBBS, HEFT-2, HEFT-1and DLS. Again, the proposed model has proven best on account of utilization.
3) As presented in Figure 22, flow time also has decreasing trend on increasing number of VM for all the strategies under experiments. LMWS-TM performs best for all configurations of VM among all strategies taken in experiments. The performance order is the same as for turnaround time i.e., LMWS-TM, SLBBS, HEFT-2, HEFT-1, and DLS. In this case, performance in the range is almost 5% to 15% on flow time for LMWS-TM over SLBBS.
4) Response time follows the same decreasing trend on varying the VMs from 8 to 128, as presented in Figure 23. Performance order is the same as in previous cases with LMWS-TM and SLLBBS at par.

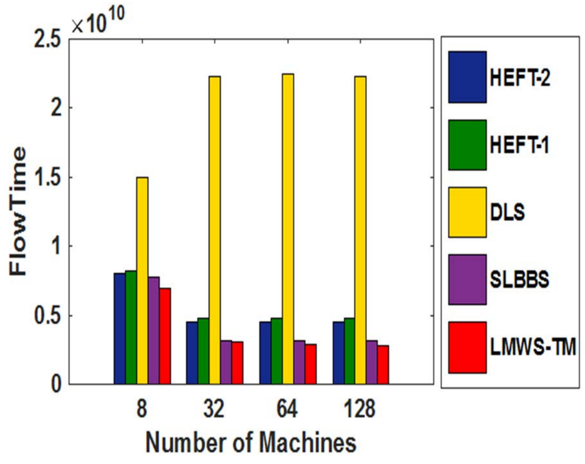

FIGURE 22. Flow time v/s number of VMs.

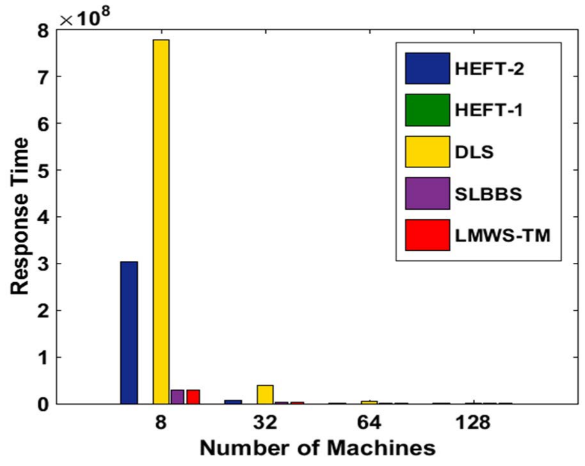

FIGURE 23. Response time v/s number of VMs.

B. STATISTICAL ANALYSIS

This part is devoted to statistical testing for performance evaluation to test the hypothesis developed from the outcomes of the simulation study. The analysis has been conducted with SPSS Statistics 20 for the data sets generated in simulation experiments and presented from Figure 12 to Figure 23 for various combinations of input parameters as explained in Section IV A, in detail. At first, the normality of data samples was tested in SPSS. Afterward, samples from the simulation study were analyzed for performance comparison of LMWS-TM, SLBBS, HEFT-2, HEFT-1, and DLS by using Wilcoxon Signed Ranks Test. The confidence interval considered in the test is 95%. The hypotheses for the normality test and comparing samples have been considered as follows:

1) Hypothesis 1 (For Normality Test)
   Ho1: The sample data are not significantly different than a normal population.
   H11: The sample data are significantly different than a normal population.
2) Hypothesis 2 (For Comparing samples)
   Ho2: There is no significant difference in various models on X.
   H12: There is a significant difference in various models on X.

where, X is a set of four parameters namely, Turnaround Time (TAT), Flow Time (FT), Average Utilization (Us) & Response Time (RT). Now, we are presenting the simulation samples in Table 8 generated by experiments for considered peers on TAT by varying batch size (number of workflows) shown in Figure 12.

TABLE 8. Simulation samples for models on varying batch size for TAT.

n_{WF}	LMWS-TM	SLBBS	HEFT-1	HEFT-2	DLS
4	5878834	6998612	54132829	31784468	1.93E+08
8	8498437	10363948	96602445	35458823	2.09E+08
16	14636065	18526665	2.03E+08	54181278	2.72E+08
32	24726553	32968738	3.89E+08	85587214	4.91E+08
64	4.75E+07	6.69E+07	8.47E+08	1.31E+08	8.91E+08
128	7.58E+07	1.17E+08	1.49E+09	1.73E+08	1.58E+09

TABLE 9. Normality tests for models for tat on varying batch size.

	Kolmogorov-Smirnov			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
TAT for LMWS-TM	.236	6	.200*	.869	6	.224
TAT for SLBBS	.252	6	.200*	.848	6	.152
TAT for HEFT-1	.255	6	.200*	.847	6	.150
TAT for HEFT-2	.208	6	.200*	.900	6	.376
TAT for DLS	.250	6	.200*	.820	6	.088

Table 9 presents the normality test results using the Kolmogorov-Smirnov and Shapiro-Wilk test for the sample given in Table 8. Here sample size is taken as 6 for each case. In the table, we can see that almost all Sig. values are greater than 0.05. Hence, the normality test null hypothesis (Ho1) is accepted, and alternate hypothesis H11 is rejected. Thus, the sample data is normally distributed for all the models for TAT on varying the batch size.

As verified, the samples are normally distributed. All the models, namely LMWS-TM, SLBBS, DLS, HEFT-1, and HEFT-2, are independent. So, the samples generated for the parameters, i.e., TAT, Us, FT, and RT, are also considered independent samples. Therefore, one way ANOVA test can be used to test the significance level between the samples.

Table 10 presents statistics of one-way ANOVA test. Here, the significant (Sig.) value is 0.014 for TAT on considered samples from various models. This Sig. value is less than 0.05 at 5% level of significance. Hence, Ho2 is rejected. Rejecting the null hypothesis proves that the proposed model i.e., LMWS-TM, significantly differs from SLBBS, DLS, HEFT-1, and HEFT-2 on TAT. The mean plot presented in Figure 24 also confirms the considerably better performance of LMWS-TM on TAT.

TABLE 10. One way ANOVA test for TAT on varying batch size.

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	1.889E+18	4	4.722E+17	3.841	0.014
Within Groups	3.074E+18	25	1.229E+17		
Total	4.963E+18	29			

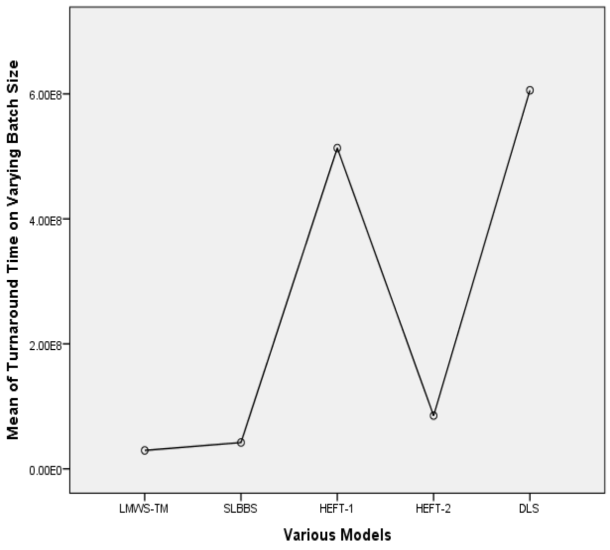

FIGURE 24. Mean plot for models on TAT.

The sample data from the remaining experiment varying batch size, depth levels, and the number of machines for TAT, Us, FT, and RT as in section IV A has been tested in a similar pattern as presented above. For all samples under study on TAT and FT, Ho2 is rejected and accepts H12. This verifies that the performance of LMWS-TM is significantly best among all cases on TAT and FT under study. Further, for the samples on RT, every time Ho2 is accepted. This is due to the almost equal RT of SLBBS and LMWS-TM for all the cases. Further, SLBBS samples are removed, and only means of four models are compared for the same (i.e., LMWS-TM, HEFT-1, HEFT-2, and DLS). In this scenario, Ho2 is rejected in all the tests. And acceptance of H12 confirms the significantly superior performance of LMWS-TM among all models excluding SLBBS on RT under study. For the case of average utilization (Us), the majority of experiments rejected the Ho2. Still, some samples under study accept the Ho2. However, the performance of LMWS-TM on Us is almost better compared to others, with some exceptions. Thus, the results of hypothesis testing confirm the conclusions drawn from the simulation study in section IV A.

V. CONCLUSION AND FUTURE SCOPE

The strategies dealing with the batch of multiple workflows perform better by considering parallelism at the workflow and task levels, satisfying the precedence constraints. This work presents a levelized multiple workflow allocation strategy with task merging (LMWS-TM) for multiple workflows having precedence constraints to optimize turnaround time. The task merging scheme is inspired by the clustering approach to reduce inter-task communication share and the total number of depth levels. It is incorporated in each workflow after partitioning, followed by allocation level-wise from first to last to enhance overall execution time. Moreover, the scheme is capable enough to use simple and flexible level attributes to tackle precedence constraints that provide the order of execution to tasks in the workflows. An experimental study has been carried out by comparing LMWS-TM with other state-of-the-art DAG scheduling strategies, viz. SLBBS, DLS, and HEFT by varying the batch size, the degree of parallelism in the workflows (depth levels), and the number of available nodes (hardware parallelism) to evaluate the suitability in the literature. The experimental study suggests that LMWS-TM performs best among all its peers significantly in all the cases under study.

AUTHOR CONTRIBUTIONS

Conceptualization: Faisal Ahmad, Mohamad Shahid; Methodology: Faisal Ahmad, Mohammad Shahid, Mohammad Sajid; Formal Analysis & Data Curation: Faisal Ahmad, Mohammad Shahid, Mahfooz Alam, Mohammad Sajid, Zubair Ashraf; Writing – Original Draft Preparation: Faisal Ahmad, Mohammad Shahid, Mahfooz Alam, Zubair Ashraf; Writing – Review & Editing: Ketan Kotecha, Gaurav Dhiman, Mohammad Sajid; Supervision: Faisal Ahmad, Mohammad Shahid, Mohammad Sajid; Funding: Ketan Kotecha, Mohammad Shahid. All authors have read and agreed to the published version of the manuscript.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

DATA AVAILABILITY STATEMENT

Data can be provided only on request.

ACKNOWLEDGMENT

(Faisal Ahmad and Mohammad Shahid contributed equally to this work.)

REFERENCES

[1] R. Buyya, C. Vecchiola, and S. T. Selvi, "Mastering cloud computing: Foundations and applications programming," Newnes, London, U.K., 2013.

[2] M. Sajid and Z. Raza, "Cloud computing: Issues & challenges," in *Proc. Int. Conf. Cloud, Big Data Trust (ICCBDT)*, Nov. 2013, pp. 35–41.

[3] R. Alsurdeh, R. N. Calheiros, K. M. Matawie, and B. Javadi, "Hybrid workflow scheduling on edge cloud computing systems," *IEEE Access*, vol. 9, pp. 134783–134799, 2021.

[4] S. Smanchat and K. Viriyapant, "Taxonomies of workflow scheduling problem and techniques in the cloud," *Future Gener. Comput. Syst.*, vol. 52, pp. 1–12, Nov. 2015.

[5] M. R. Gary and D. S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness," Freeman, New York, NY, USA, 1979.

[6] S. Sharma and M. Sajid, "Integrated fog and cloud computing issues and challenges," *Int. J. Cloud Appl. Comput.*, vol. 11, no. 4, pp. 174–193, Oct. 2021.

[7] R. A. Haidri, M. Alam, M. Shahid, S. Prakash, and M. Sajid, "A deadline aware load balancing strategy for cloud computing," *Concurrency Comput., Pract. Exper.*, vol. 34, no. 1, p. e6496, Jan. 2022.

[8] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Gener. Comput. Syst.*, vol. 29, no. 3, pp. 682–692, 2013.

[9] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the Kepler system," *Concurrency Comput. Pract. Exper.*, vol. 18, no. 10, pp. 1039–1065, 2006.

[10] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[11] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed data-parallel programs from sequential building blocks," *ACM SIGOPS Oper. Syst. Rev.*, vol. 41, no. 3, pp. 59–72, 2007.

[12] K. Svitil, "Gravitational waves detected 100 years after Einstein's prediction: Astronomy," *Quest*, vol. 12, no. 2, pp. 24–26, 2016.

[13] B. Saovapakhiran, G. Michailidis, and M. Devetsikiotis, "Aggregated-DAG scheduling for job flow maximization in heterogeneous cloud computing," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2011, pp. 1–6.

[14] R. Khorsand, F. Safi-Esfahani, N. Nematbakhsh, and M. Mohsenzade, "Taxonomy of workflow partitioning problems and methods in distributed environments," *J. Syst. Softw.*, vol. 132, pp. 253–271, Oct. 2017.

[15] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 8, p. e4041, Apr. 2017.

[16] U. Hönig and W. Schiffmann, "A meta-algorithm for scheduling multiple dags in homogeneous system environments," in *Proc. 18th IASTED Int. Conf. Parallel Distrib. Comput. Syst. (PDCS)*, 2006, pp. 1–6.

[17] M. H. Hilman, M. A. Rodriguez, and R. Buyya, "Multiple workflows scheduling in multi-tenant distributed systems: A taxonomy and future directions," *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–39, 2021.

[18] Y.-K. Kwok and I. Ahmad, "Benchmarking the task graph scheduling algorithms," in *Proc. 1st Merged Int. Parallel Process. Symp. Symp. Parallel Distrib. Process.*, 1998, pp. 531–537.

[19] L. Zhu, Z. Sun, W. Guo, Y. Jin, W. Sun, and W. Hu, "Dynamic multi DAG scheduling algorithm for optical grid environment," *Proc. SPIE*, vol. 6784, Nov. 2007, Art. no. 67841F.

[20] Y. Wang, C. Jia, and Y. Xu, "Multiple DAGs dynamic workflow scheduling based on the primary backup algorithm in cloud computing system," in *Proc. 9th Int. Conf. Broadband Wireless Comput., Commun. Appl.*, Nov. 2014, pp. 177–182.

[21] S. Sharif, J. Taheri, A. Y. Zomaya, and S. Nepal, "Online multiple workflow scheduling under privacy and deadline in hybrid cloud environment," in *Proc. IEEE 6th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2014, pp. 455–462.

[22] H. Zhao and R. Sakellariou, "Scheduling multiple DAGs onto heterogeneous systems," in *Proc. 20th IEEE Int. Parallel Distrib. Process. Symp.*, Apr. 2006, pp. 14–28.

[23] M. Shahid and Z. Raza, "Level-based batch scheduling strategies for computational grid," *Int. J. Grid Utility Comput.*, vol. 5, no. 2, pp. 135–148, 2014.

[24] Y.-K. Kwok and I. Ahmad, "Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, no. 5, pp. 506–521, May 1996.

[25] J.-J. Hwang, Y.-C. Chow, F. D. Anger, and C.-Y. Lee, "Scheduling precedence graphs in systems with interprocessor communication times," *SIAM J. Comput.*, vol. 18, no. 3, pp. 244–257, 1989.

[26] G. C. Sih and E. A. Lee, "A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 2, pp. 175–187, Feb. 1993.

[27] 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.

[28] J. Chen, C. Du, P. Han, and X. Du, "Work-in-progress: Non-preemptive scheduling of periodic tasks with data dependency upon heterogeneous multiprocessor platforms," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2019, pp. 540–543.

[29] T. Li, D. Cao, Y. Lu, T. Huang, C. Sun, Q. Dong, and X. Gong, "DBEFT: A dependency-ratio bundling earliest finish time algorithm for heterogeneous computing," *IEEE Access*, vol. 7, pp. 173884–173896, 2019.

[30] M. Alam, M. Shahid, and S. Mustajab, "SAHEFT: Security aware heterogeneous earliest finish time workflow allocation strategy for IaaS cloud environment," in *Proc. IEEE Madras Sect. Conf. (MASCON)*, Aug. 2021, pp. 1–8.

[31] M. Shahid, M. Alam, F. Hasan, and M. Imran, "Security-aware workflow allocation strategy for IaaS cloud environment," in *Proc. Int. Conf. Commun. Comput. Technol. Algorithms Intell. Syst.*, S. Purohit, D. S. Jat, R. Poonia, S. Kumar, and S. Hiranwal, Eds. Singapore: Springer, 2021, pp. 241–252.

[32] M. Shahid, Z. Ashraf, M. Alam, F. Ahmad, and M. Imran, "A multi-objective workflow allocation strategyin IaaS cloud environment," in *Proc. Int. Conf. Comput., Commun., Intell. Syst (ICCCIS)*, Feb. 2021, pp. 308–313, doi: 10.1109/ICCCIS51004.2021.9397081.

[33] M. C. Calzarossa, M. L. D. Vedova, L. Massari, G. Nebbione, and D. Tessera, "Multi-objective optimization of deadline and budget-aware workflow scheduling in uncertain clouds," *IEEE Access*, vol. 9, pp. 89891–89905, 2021.

[34] S. Darbha and D. P. Agrawal, "Optimal scheduling algorithm for distributed-memory machines," *IEEE Trans. Parallel Distrib. Syst.*, vol. 9, no. 1, pp. 87–95, Jan. 1998.

[35] D. Nurmi, A. Mandal, J. Brevik, C. Koelbel, R. Wolski, and K. Kennedy, "Evaluation of a workflow scheduler using integrated performance modelling and batch queue wait time prediction," in *Proc. ACM/IEEE SC Conf. (SC)*, Nov. 2006, p. 29.

[36] O. Sonmez, N. Yigitbasi, S. Abrishami, A. Iosup, and D. Epema, "Performance analysis of dynamic workflow scheduling in multicluster grids," in *Proc. 19th ACM Int. Symp. High Perform. Distrib. Comput. (HPDC)*, 2010, pp. 49–60.

[37] Z.-F. Yu and W.-S. Shi, "Queue waiting time aware dynamic workflow scheduling in multicluster environments," *J. Comput. Sci. Technol.*, vol. 25, no. 4, pp. 864–873, Jul. 2010.

[38] J. Chen, Y. He, Y. Zhang, P. Han, and C. Du, "Energy-aware scheduling for dependent tasks in heterogeneous multiprocessor systems," *J. Syst. Archit.*, vol. 129, Aug. 2022, Art. no. 102598.

[39] M. A. Iverson, F. Özgüner, and G. J. Follen, "Parallelizing existing applications in a distributed heterogeneous environment," in *Proc. 4th Heterogeneous Comput. Workshop (HCW)*, 1995, pp. 93–100.

[40] L.-T. Lee, C.-W. Chen, H.-Y. Chang, C.-C. Tang, and K.-C. Pan, "A non-critical path earliest-finish algorithm for inter-dependent tasks in heterogeneous computing environments," in *Proc. 11th IEEE Int. Conf. High Perform. Comput. Commun.*, 2009, pp. 603–608.

[41] F. Wu, Q. Wu, and Y. Tan, "Workflow scheduling in cloud: A survey," *J. Supercomput.*, vol. 71, no. 9, pp. 3373–3418, 2015.

[42] S. Kumar and Z. Raza, "Using clustering approaches for response time aware job scheduling model for Internet of Things (IoT)," *Int. J. Inf. Technol.*, vol. 9, no. 2, pp. 177–195, Jun. 2017.

[43] B. Kruatrachue and T. Lewis, "Grain determination for parallel processing systems," in *Proc. 21st Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 2, Jan. 1988, pp. 119–120.

[44] B. Shirazi, H.-B. Chen, and J. Marquis, "Comparative study of task duplication static scheduling versus clustering and non-clustering techniques," *Concurrency, Pract. Exper.*, vol. 7, no. 5, pp. 371–389, Aug. 1995.

[45] D. Kim and J. Park, "Two-way dominant sequence clustering for processor scheduling," *Inf. Process. Lett.*, vol. 49, no. 4, pp. 203–208, Feb. 1994.

[46] B. Jedari and M. Dehghan, "Efficient DAG scheduling with resource-aware clustering for heterogeneous systems," in *Computer and Information Science*. Berlin, Germany: Springer, 2009, pp. 249–261.

[47] H. Kanemitsu, M. Hanada, and H. Nakazato, "Clustering-based task scheduling in a large number of heterogeneous processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 11, pp. 3144–3157, Nov. 2016.

[48] L. F. Bittencourt and E. R. M. Madeira, "Towards the scheduling of multiple workflows on computational grids," *J. Grid Comput.*, vol. 8, no. 3, pp. 419–441, Sep. 2010.

[49] A. Hirales-Carbajal, A. Tchernykh, R. Yahyapour, J. L. González-García, T. Röblitz, and J. M. Ramírez-Alcaraz, "Multiple workflow scheduling strategies with user run time estimates on a grid," *J. Grid Comput.*, vol. 10, no. 2, pp. 325–346, Jun. 2012.

[50] W. Jing and Y. Liu, "Multiple DAGs reliability model and fault-tolerant scheduling algorithm in cloud computing system," *Comput. Model. Technol.*, vol. 18, no. 8, pp. 22–30, 2014.

[51] W. Chen, Y. C. Lee, A. Fekete, and A. Y. Zomaya, "Adaptive multiple-workflow scheduling with task rearrangement," *J. Supercomput.*, vol. 71, no. 4, pp. 1297–1317, Apr. 2015.

[52] K. Bochenina, N. Butakov, and A. Boukhanovsky, "Static scheduling of multiple workflows with soft deadlines in non-dedicated heterogeneous environments," *Future Gener. Comput. Syst.*, vol. 55, pp. 51–61, Feb. 2016.

[53] M. Sajid and Z. Raza, "Energy-aware stochastic scheduler for batch of precedence-constrained jobs on heterogeneous computing system," *Energy*, vol. 125, pp. 258–274, Apr. 2017.

[54] M. Sajid and Z. Raza, "Energy-aware stochastic scheduling model with precedence constraints on DVFS-enabled processors," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 24, pp. 4117–4128, 2016.

[55] M. Shahid and Z. Raza, "A precedence based load balancing strategy for batch of DAGs for computational grid," in *Proc. Int. Conf. Contemp. Comput. Informat. (ICI)*, Nov. 2014, pp. 1289–1295.

[56] M. Shahid and Z. Raza, "Performance evaluation of static level based batch scheduling strategy (SLBBS) for computational grid," in *Proc. Int. Conf. Parallel, Distrib. Grid Comput.*, Dec. 2014, pp. 169–175.

[57] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, and H. Xie, "Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 39974–39982, 2019.

FAISAL AHMAD received the M.C.A. degree from Aligarh Muslim University, in 2009, and the Ph.D. degree in software engineering from the NIT Durgapur, India, in 2019. He is currently a Senior Software Engineer at Workday Inc., USA. He has 14 years of industrial experience as different roles from developer to architect role. He has published papers in international journals along with many international and national conference. His current research interests include cloud computing, software engineering, machine learning, and portfolio optimization. He is also a reviewer in many international journals.

MOHAMMAD SHAHID (Member, IEEE) received the M.Tech. and Ph.D. degrees from the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India, and the M.C.A. degree from the Department of Computer Science, Aligarh Muslim University, Aligarh, India. He is currently an Assistant Professor of computer science with the Department of Commerce, Aligarh Muslim University. He has published many research papers in international conferences and international journals of repute, including Elsevier, Springer, InderScience, Wiley, Taylor & Francis, and IGI Global. His research interests include grid/cloud computing, workflow scheduling, evolutionary computing, meta-heuristic optimization, and portfolio optimization. He was awarded a Research Startup Grant from the UGC-BSR, India, in 2017.

MAHFOOZ ALAM received the B.C.A. and M.C.A. degrees in computer applications from Indira Gandhi National Open University (IGNOU), New Delhi, India, and the M.Tech. degree in computer science and engineering from Dr. A. P. J. Abdul Kalam Technical University, Lucknow, India, in 2015. He has also served for six years as an Assistant Professor with the Department of Computer Science, Al-Barkaat College of Graduate Studies (ABCGS), Aligarh, India. He is currently a Research Scholar with the Department of Computer Science, Aligarh Muslim University (AMU), Aligarh. He has published many research papers in international conferences and international journals of repute. His research interests include parallel and distributed computing, cloud computing, workflow scheduling, and load balancing.

ZUBAIR ASHRAF received the Ph.D. degree in computer science from South Asian University, New Delhi, India, in February 2020. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Koneru Lakshmaiah University, Andhra Pradesh, India. His research interests include multi-criteria decision making, evolutionary optimization, nature-inspired intelligent computation, deep learning, and fuzzy systems. He is the IEEE Young Professional and an Active Member of several societies, including the IEEE Computational Intelligence Society and EUSFLAT. He is also an Active Reviewer of journals, such as the IEEE Transactions on Fuzzy System, *Soft Computing*, *Applied Soft Computing*, *Applied Mathematics*, and the *International Journal of Intelligence System*.

MOHAMMAD SAJID received the M.C.A., M.Tech., and Ph.D. degrees from the School of Computer and Systems Sciences, Jawaharlal Nehru University (JNU), New Delhi. He is currently working as an Assistant Professor with the Department of Computer Science, Aligarh Muslim University, India. He has published many research papers in international conferences and international journals of repute. His research interests include parallel and distributed computing, cloud computing, evolutionary and swarm intelligence algorithms, and combinatorial optimization problems. He was awarded a Research Startup Grant (ten lakhs) from UGC-BSR, India, in 2017.

KETAN KOTECHA is currently an Administrator and a Teacher of deep learning. He has expertise and experience of cutting-edge research and projects in AI and deep learning for over last 25 years. He has published three patents and delivered keynote speeches at various national and international forums, including at Machine Intelligence Labs, USA, the IIT Bombay under the World Bank Project, the International Indian Science Festival organized by the Department of Science and Technology, Government of India, and many more. He has published widely with more than 100 publications in several excellent peer-reviewed journals on various topics ranging from cutting edge AI, education policies, teaching learning practices, and AI for all. His research interests include artificial intelligence, computer algorithms, machine learning, and deep learning. He was a recipient of the two SPARC projects in AI and worth INR 166 lacs from MHRD, Government of India, in collaboration with Arizona State University, USA; and the University of Queensland, Australia. He was also a recipient of numerous prestigious awards, such as Erasmus + Faculty Mobility Grant to Poland, DUO-India Professors Fellowship for research in responsible AI in collaboration with Brunel University, U.K.; LEAP Grant at Cambridge University, U.K.; UKIERI Grant with Aston University, U.K.; and a Grant from the Royal Academy of Engineering, U.K., under Newton Bhabha Fund. He is also an Academic Editor of *PeerJ Computer Science* journal and an Associate Editor of IEEE Access journal.

GAURAV DHIMAN (Senior Member, IEEE) received the master's degree in computer applications and the Ph.D. degree in computer engineering from the Thapar Institute of Engineering and Technology, Patiala. He is currently working as an Assistant Professor with the Department of Computer Science, Government Bikram College of Commerce, Patiala. He is also an Adjunct Faculty with Chandigarh University and Graphic Era Deemed to be University. He has published more than 200 peer-reviewed research articles (indexed in SCI/SCIE) and ten international books. He was selected as an Outstanding Reviewer from *Knowledge-Based Systems* (Elsevier). He is also serving as the guest editor of more than 40 special issues in various peer-reviewed journals. More information can be found at: http://www.dhimangaurav.com.