

Received September 23, 2019, accepted October 22, 2019, date of publication October 29, 2019,  
date of current version November 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2950110

# A Management System for Servicing Multi-Organizations on Community Cloud Model in Secure Cloud Environment

KALKA DUBEY<sup>1</sup>, MAHMOUD Y. SHAMS<sup>2</sup>, S. C. SHARMA<sup>1</sup>, ABDULAZIZ ALARIFI<sup>3</sup>,  
MOHAMMED AMOON<sup>1,3,4</sup>, AND AIDA A. NASR<sup>1,2</sup>

<sup>1</sup>Cloud Computing and Wireless Sensor Lab, IIT Roorkee, Roorkee 247001, India

<sup>2</sup>Faculty of Artificial Intelligence, Kafrelsheikh University, Kafr el-Sheikh 33511, Egypt

<sup>3</sup>Department of Computer Science, Community College, King Saud University, Riyadh 11437, Saudi Arabia

<sup>4</sup>Department of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufia University, Menouf 32952, Egypt

Corresponding author: Mohammed Amoon (mamoon@ksu.edu.sa)


The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through Research Group No (RG-1440-039) and also thank the IIT Roorkee India.

**ABSTRACT** The emergence of cloud computing has been growing rapidly in the last decades especially for workflow scheduling. Organizations with the same requirements and needs go to use the community cloud for saving costs. One of the important challenges of using the community cloud is resource allocation and task scheduling. In this paper, we propose a new Management System for servicing Multi-organizations in a Community cloud (MSMC) in a secure cloud environment. The MSMC employs a virtual machine allocation algorithm to organize the community cloud usage among the organizations, where it allocates the available virtual machines according to the use of each organization in an efficient and fair way to execute the submitted applications. Moreover, the MSMC system proposes a new scheduling algorithm, called Ideal Distribution Algorithm (IDA), to schedule the workflow tasks to the virtual machines of the cloud considering both the deadline and cost constraints. Additionally, an enhanced version of the IDA, called Enhanced IDA (EIDA) is proposed to provide load balancing required by the cloud. The simulation experiments show that the system can improve the system ability under deadline constraints and improve the monetary cost.

**INDEX TERMS** Management system, community cloud, resource allocation, task scheduling, load balance.

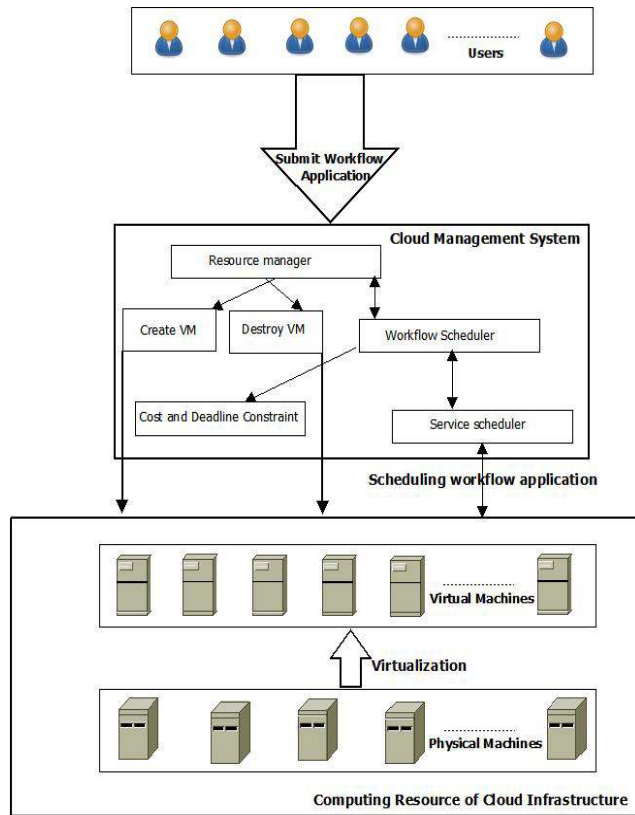
## I. INTRODUCTION

In recent years, data processing demand has been increased for various disciplines such as geographical science, engineering, business, financial, educational, and healthcare applications. Cloud computing has been exceedingly recognized as a capable solution for data processing. It is a high-performance computing model that delivers its services through the Internet and executes large scientific applications. Cloud computing can deliver three main types of services namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [1], [2]. IaaS cloud provides massive computing hardware infrastructure platform and software resources in the form of services to the cloud users. PaaS cloud provides a platform where users

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

can deploy their applications and use the exiting platform for constructing their application while in SaaS cloud users can only run an application on cloud infrastructure over the Internet.

Cloud computing models are private, public, community and hybrid models. The Community cloud model is important for organizations that have the same requirements. However, the employment of a bad management system leads to a decrease in the performance of executing the submitted applications/workflows. Workflow is a very popular method of modeling high data processing applications when executed in cloud computing environments. Workflow represented by a Direct Acyclic Graph (DAG) [3] in which, graph nodes denote the computational tasks and graph edges represent the dependencies among the graph's tasks. The size of the DAG depends on the scientific application. The workflow size is considered small if the scientific application



**FIGURE 1.** Workflow scheduling system architecture on the community cloud.

is easy and less complex. Otherwise, the workflow size is vast [4].

Workflow scheduling is a process, which performs the mapping of workflow's tasks on the heterogeneous and distributed resources of a computing system. A suitable number of resources should be allocated for the execution of the tasks of a workflow so that it can be completed with satisfying the user-defined constraints. Figure 1 illustrates the common architecture used for scheduling workflows on community cloud systems. The architecture has three components: cloud user interface, cloud management system and computing resources. The cloud user interface accepts the workflow applications with requirements and constraints from users and passes them to the following component in the architecture. The cloud management system receives multiple (Quality of Service) QoS requirements and user-defined constraints that are used by the workflow scheduler [5], [6]. Constraints will include deadlines, cost, resource utilization, etc. The main function of the resource manager is to perform resource mapping of workflow tasks based on the objective function that comprises many constraints. Computing resource is the third component of the architecture, which consists of the virtual infrastructure layer and the physical infrastructure layer. A mapping process is also performed between the virtual and the physical infrastructure layer. Physical resources provided as virtual resources to the workflow applications.

Workflow scheduling represents one of the most prominent and challenging issues in the cloud computing model. The scheduling problem is considered as NP-Complete [7] and several heuristic algorithms like Heterogeneous Earliest Finish Time (HEFT), Min-Min, Max-Min have been found in the literature for solving it.

The main contributions of this research work are:

- Proposing an architecture for the community cloud computing.
- Proposing a new management system for serving multi-organizations in the community cloud in a secure cloud environment. It has the ability to:
  - Managing resource allocation in an efficient way.
  - Scheduling submitted workflow on the available resources to minimize the makespan and the cost under the deadline constraints.
  - Improving the load balance and utilization system.
- Evaluating the performance of the proposed management system using CloudSim environment with different types of well-known workflows.

The remaining of the paper is divided as follows: Section 2 gives a brief overview of the related research work. In section 3, the system and application model are defined. Section 4 explains the proposed MSMC system in detail. In Section 5, experimental results and discussions are provided. Finally, section 6 presents the conclusion and future work of this study.

## II. RELATED WORK

Workflow scheduling is a crucial challenge in cloud computing [8]. Several works of research have been done to deal with the workflow scheduling problem. The [9], [10] focused on the minimization of the execution time of workflow while in [11], a heuristic approach is used to minimize the total execution time of the workflow. In this heuristic approach, information is needed in advance for deciding on which task is scheduled on which resource and this task is scheduled on the selected resource before runtime. HEFT [12] is one of the well-known heuristics for solving the scheduling problem of workflows in heterogeneous cloud computing resources. In the HEFT, tasks are sorted according to rank values and scheduled one by one to resources, which minimizes its completion time while considering the data transmission time. It also minimizes the makespan and generates a fair schedule with very low complexity.

Researchers [13], [14] extended the HEFT to solve the multi-objective workflow scheduling problem. For optimization of time under the budget constraints, the HEFT employed with GAIN and LOSS approach [15]. The HEFT generated the optimal schedule with a very low time complexity and also managed the very low space complexity. It has been widely combined with many other heuristic approaches [16], [17]. To choose the best configuration for the customer, Pietri and Sakellariou [13] combine the CSFS-MAX and CSFS-MIN methods for generating the initial schedule in the

HEFT technique while the HEFT works as a makespan work scheduler in [18], [19].

Mao and Humphrey [20] presented a scaling consolidation scheduling (SCS) algorithm for the execution of the workflow in cloud environments. The SCS algorithm is based on the concept of instance consolidation and bundling of the task. It is mainly focused on the heterogeneity of virtual machines and the enhancement of the performance by reducing the acquisition delay of virtual machines. However, this algorithm did not consider the time of transferring data between tasks which increased the overall cost and makespan time to complete the workflow. Malawski et al. [21] presented static and dynamic algorithms in the cloud while considering the deadline and cost constraints. They presented three algorithms named Static Provisioning Static Scheduling (SCSS), Dynamic provisioning Dynamic scheduling (DPSS) and Workflow Aware Dynamic Provisioning Dynamic Scheduling (WA-DPSS). These algorithms try to maximize the performance by reducing the cost and makespan under the consideration of the deadline of the workflow.

Other similar works presented in [22]–[24] are based on the critical path. Abrishami et al. [22] discussed the static algorithm for the IaaS cloud termed as IaaS Cloud Partial Critical Path (IC-PCP) algorithm. It starts with estimating the latest finish time of each task and then determines the particle critical path (PCP) of each node in the DAG. Then, each task is scheduled for the cheapest VM instance that can execute it before the estimated finish time. If there is no suitable VM instance, which does not meet the required finish time constraints of the task, new VM instances are created and the task is assigned to this newly created VM which can complete it within the limits of finish time constraints.

Authors in [25] have considered the scheduling conflicts caused by similar tasks. Their scheduling method is based on using the backfilling algorithm for deadline-based tasks. Also, they have used three methods of multi-criteria of making decisions to deal with the scheduling conflicts.

Also, the work in [26] is based on using the backfilling algorithm. In this time, it is used to allocate resources to the deadline sensitive lease tasks. The work has addressed some issues of the backfilling algorithm such as conflicts of similar leases and scheduling of new coming tasks during the execution.

### III. MODELS

#### A. WORKFLOW MODEL

A workflow scheduling problem can be represented by a Direct Acyclic Graph, DAG  $(V, E)$  where  $V$  is the set of tasks  $\{t_1, t_2, \dots, t_n\}$  and  $E$  is the set of edges or dependencies among the tasks. Every single task  $t_i$  in the DAG represents an individual application with different workloads and each edge or dependency like  $e(i, j) = (t_i, t_j)$  indicates that the execution of task  $t_j$  cannot start before the completion of task  $t_i$  execution. So, task  $t_i$  is denoted as a parent node

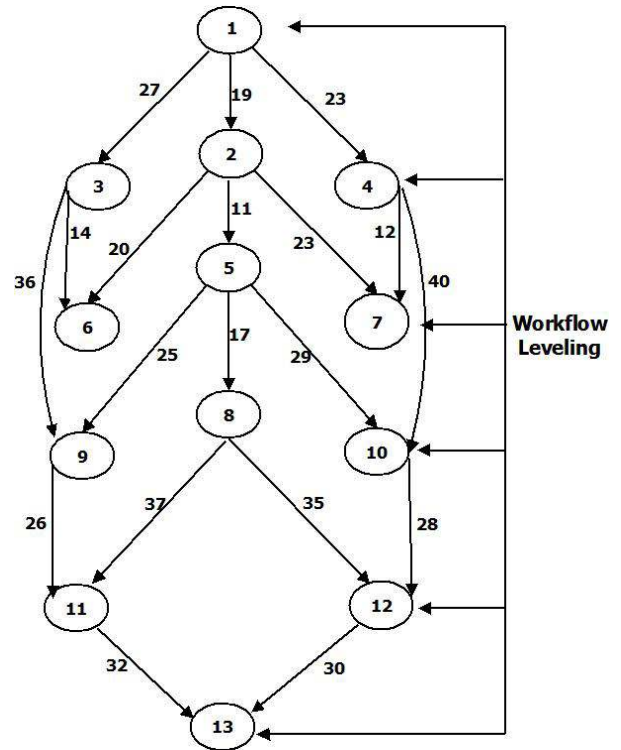


FIGURE 2. DAG sample example.

and task  $t_j$  is denoted as a child node. Furthermore, data transmission  $d_{i,j}$  is attached with the edge  $e_{i,j}$ . Each task can have multiple parent nodes and also it can have multiple child nodes. A sample of DAG is displayed in figure 2. A node with no parents is called the entry node and the node without children is called the exit node.

A workflow scheduler can be modeled as  $\langle R, T, M \rangle$  where  $R$  is the set of resources,  $T$  is the set of dependent tasks and  $M$  is the set of mappings between tasks and resources. Each mapping  $m$  reveals that the task  $t_i$  is allocated to the resource  $r_j$  for the time interval starting from time  $S_{i,j}$  and ending at time  $F_{i,j}$ . The cost ( $t_i$ ) is the cost of a single task  $t_i$ .

$$R = \{r_1, r_2, \dots, r_x\}. \tag{1}$$

$$T = \{t_1, t_2, \dots, t_y\}. \tag{2}$$

$$M = \{m_1, m_2, \dots, m_z\}. \tag{3}$$

$$m = \{t_i, r_j, S_{i,j}, F_{i,j}\}. \tag{4}$$

The execution time or makespan of a workflow can be defined as:

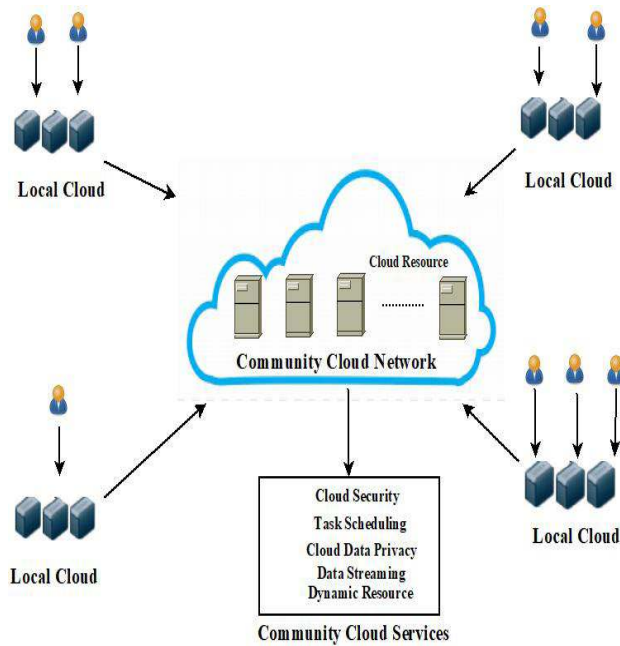
$$f(R, M) = \max_{i,j} \{F_{i,j}\}. \tag{5}$$

The total cost of scheduling can be obtained as:

$$c(R, M) = \sum_{i=1}^n EC_i, \tag{6}$$

where  $EC_i$  is defined for each mapping as:

$$EC_i = (\text{ceil}(F_{i,j} - S_{i,j}) \times \text{cost}(t_i)). \tag{7}$$



**FIGURE 3.** Community cloud network with local cloud and community cloud services.

The constraint specified by the user is denoted as  $d$  and it can be defined for optimization problem of workflow scheduling as:

$$\min\{c(R, M), f(R, M)\} < d. \quad (8)$$

### B. COMMUNITY CLOUD MODEL

Community Cloud [27] system is designed for fulfilling the requirement of a particular group of users that formed a community. Public cloud offers its services through the Internet to anyone who has valid credentials while private cloud is restricted for specific user sets like organizations, institutes, research panels, small companies, business firms, etc. Community cloud bridges the big gap between the public cloud and the private cloud [28]. Community cloud can have different types and forms of resources based on the Community member-specific requirement, services and characteristics.

Figure 3 indicates the scenario of a community cloud network and its services. The figure shows the local cloud which has hosts, resources and cloud data centers connected via community cloud network for provisioning of exclusive use of a community of users with shared concern, interests, common problems and solutions managed by the community. Community services provided by the cloud provider aim to provide a high level of security to secure the local cloud user data from unauthorized users. Task scheduling and dynamic resource pooling service are also provided by the community cloud.

### C. SECURE CLOUD FRAMEWORK

We have proposed a new secure cloud framework, as shown in figure 4, to provide extra security to the cloud user data throughout the whole data transaction performed in the cloud

computing model. We have classified the security framework in three phases namely authentication or log in phase, authorization or permission access phase and processing phase. The first phase ensures that the only legislative users access the cloud services and no malicious or unauthenticated users can access the cloud services. After receiving the request from the user, the cloud service provider checks that the request coming from authenticated or legislative users using the fingerprint module. If the request was issued from a legislative user, then the access permission granted to that user in the second phase. Otherwise, the user is blocked and the necessary action is taken for restricting the entry of malicious users in the cloud computing network. After authorization, user data is encrypted by applying an encryption algorithm and is sent to the CSP through the unsecured network or the Internet. CSP received the encrypted data and performed decryption by applying a decryption algorithm. In the last phase, all the data related processing is performed.

Advanced Encryption Standard (AES) algorithm [29] is used for the encryption of the user's plain text to cipher text and the reverse of AES is used to convert the cipher text to plain text. The Fingerprint module performs the authentication part in secure cloud framework. This module includes the fingerprint registration with the matching of fingerprints. Fingerprint registration is the process of extracting fingerprint image, value, characteristics from the legislative user and store in the fingerprint database while in the fingerprint matching process fingerprint server is used. Fingerprint matching can be performed in two parts: first is the validation and second is the identification.

### IV. THE PROPOSED MANAGEMENT SYSTEM

The proposed system employs the community cloud model that comprises a set of organizations. One of the main challenges facing the management system of the community cloud is the scheduling, where high response time is required when the employers of all organizations submit their applications at the same time. At this time, the scheduler of the cloud cannot efficiently manage available resources.

The proposed management system comprises three algorithms: allocation algorithm, IDA algorithm and Enhanced IDA (EIDA) algorithm. The main aim of the system is to manage the execution of tasks of workflows on the shared resources of community clouds.

The allocation algorithm operates at the level of virtual machines to allocate the required number of virtual machines to each organization in the cloud. Thereafter, the scheduling algorithm, denoted as IDA, assigns each workflow's task to the suitable virtual machine considering both the time and cost deadlines required. For better balancing of the workload, the EIDA algorithm provides improved scheduling considering level load balancing (LLB). The algorithm tries to distribute the load in levels so that the children nodes of that level could not cause the delay in execution and also can balance the execution time of tasks of each level.



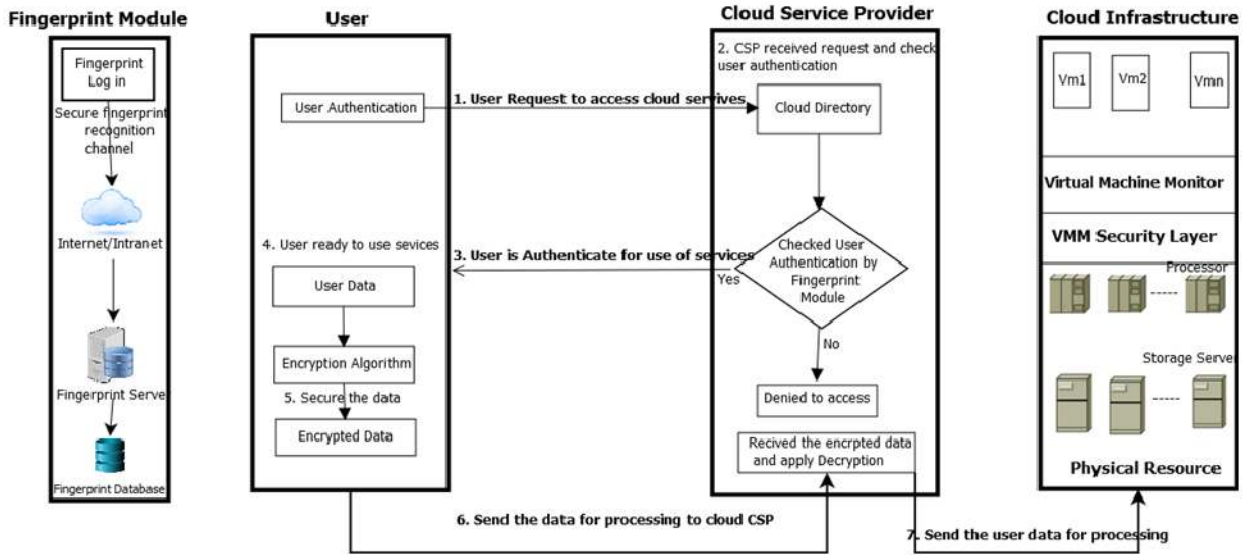


FIGURE 4. Secure framework for MSMC in the cloud environment.

The following subsections explain the details of the algorithms of the proposed system.

**A. VM ALLOCATION ALGORITHM**

In the proposed system, we have developed a new allocation algorithm that efficiently allocates the available VMs to serve multiple organizations with multi-employees. The algorithm has three steps:

Step (1): In this step, the system equally divides the resources among the organizations into  $v$  virtual machines for each organization according to:

$$v = m/o, \tag{9}$$

where  $m$  is the number of available VMs and  $o$  is the number of organizations. Each organization will have the same number of VMs. Every organization with no submitted workflows will add its VMs to the Free-List of virtual machines.

Step (2): According to the number of workflow tasks for each organization, the system adds an extra number of virtual machines to the organization from the Free-List as follows:

$$EV = \begin{cases} \frac{f}{w} & f > w \\ \frac{f}{a} & f = a, \end{cases} \tag{10}$$

where  $EV$  is the number of extra VMs,  $f$  is the number of VMs in the Free-List,  $w$  is the number of workflows and  $a$  is the number of large workflows.

Step (3): In this step, the management system checks the newly submitted workflows. If it finds newly submitted workflows, two scenarios are applied:

1. If there are free VMs, the system allocates VMs from the Free-List to the newly submitted workflows.

**Algorithm 1** VM Allocation Algorithm

**Input:**  $m$  is the number of available VMs.  
 $o$  is the number of organizations.

**Output:** allocation of VMs to  $o$

- 1-  $v = m/o$ ;
- 2- **For** each organization  $o$  **do**
- 3-   Allocate  $v$  VMs to  $o$ ;
- 4-   Calculate  $EV$  as in Eq. (10);
- 5-   Add  $EV$  to  $o$ ;
- 6- **EndFor**
- 7- **For** each organization  $o$  **do**
- 8-   **If** there are newly submitted workflows **then**
- 9-    **If** the number of VMs allocated to  $o < v$  **then**
- 10-     **If** Free-List is not empty **then**
- 11-       Allocate  $EV$  VMs from the Free-List to  $o$ ;
- 12-     **Else**
- 13-       Get the organizations which have a number of VMs  $> v$  and a smallest number of tasks;
- 14-       Compute  $z$ ; //Eq. (11)
- 15-       Interrupt  $z$  VMs to free them;
- 16-       Allocate the free VMs  $z$  to the new workflows;
- 17-     **EndIf**
- 18-    **EndIf**
- 19-    **EndIf**
- 20- **EndFor**

2. If there are no free VMs, the system interrupts  $z$  virtual machines from organization that have a number of VMs greater than  $v$  and have a small number of workflows and allocates them to the newly submitted workflows. We can define  $z$  by:

$$z = \sum_{i=1}^o EV. \tag{11}$$

## B. IDA SCHEDULING ALGORITHM

This section explains a heuristic-based workflow scheduling algorithm that considers cost and deadline constraints. The algorithm has three steps:

Step (1): Transforming Workflow into DAG.

Workflows are used to manage large scientific computations. So, this paper assumes that any workflow will be transformed in the form of a DAG as described in Section 3.

Step (2): Leveling.

In this step, all tasks in the DAG are grouped into separated sets called levels and each level is defined by a number. The tasks of the same level show no dependency between them and thus each level could be considered as a set of level tasks (SLT). The level of an individual task is an integer positive number and it indicates the task belongs to which SLT. Figure 3 shows a DAG and each task in the DAG is associated with a particular level. In this figure, the tasks are grouped into six levels. Task 1 represents the first level. Tasks 2, 3 and 4 represent the second level. The third level contains tasks 5, 6 and 7. The fourth level contains tasks 8, 9 and 10. The fifth level contains tasks 11 and 12. Task 13 represents the sixth level.

---

### Algorithm 2 The IDA Scheduling Algorithm

---

**Input:** A workflow

**Output:** A schedule

---

1. Divide the workflow into levels;
  2. Sort the levels according to the dependency order;
  3. Assign the task in level 1 to the VM with the minimum earliest finish time;
  4. Remove level 1 from the list of levels;
  5. **For** each level **do**
  6. Find  $e = \text{no. of edges}$  for each task in the level;
  7. Sort the tasks in descending order according to  $e$ ;
  8. **For** each task  $i$  in the sorted list **do**/allocation phase
  9. Find the  $VM_j$  with the minimum cost ( $i$ );
  10. **If**  $FT(T_i, VM_j) \leq \text{Deadline}$  **then**
  11. Assign  $i$  to  $VM_j$ ;
  12. **Else**
  13. Find the  $VM_j$  which gives the minimum  $(\text{cost}(i) * ET(i))$ ;
  14. **EndIf**
  15. **If**  $FT(T_i, VM_j) \leq \text{Deadline}$  **then**
  16. Assign  $i$  to  $VM_j$ ;
  17. **Else**
  18. Find the  $VM_j$  which gives the minimum  $(ET(i))$ ;
  19. Assign  $i$  to  $VM_j$ ;
  20. **EndIf**
  21. **EndFor**
  22. **EndFor**
- 

Step (3): Scheduling.

During this step of the IDA algorithm, the workflow tasks are distributed in an efficient way, where it can increase the balancing degree and minimize the cost considering the

deadline constraints. This step consists of two phases: the preparation phase and the allocation phase.

In the preparation phase, the required essential information about the workflow and the resources are prepared. Then, the workflow levels are sorted according to the dependency order, where all tasks in level 1 should be at the top. Finally, the IDA algorithm sorts the tasks of each level according to the number of edges by descending order. We consider the number of edges to give each task a specific priority because it has a vital role in executing the task early to reduce communication time. Wherever the task with more number of edges is executed early, the overall communication time will decrease.

In the allocation phase, the IDA allocates the tasks of the levels starting from level one. Firstly, it selects a task from the sorted list of the level. Then, the IDA algorithm assigns the tasks of the level into the virtual VM which gives the minimum cost. After finishing the assignment of all tasks in a level, it will transfer to the next level. For each level, the algorithm tries to assign tasks to inexpensive VMs considering deadline constraints. The main purpose of this phase is to allocate the workflow tasks by minimizing the cost taking into consideration deadline constraints.

The IDA algorithm selects a task and searches for the VM of the minimum cost. Next, the algorithm calculates the finish time of task  $i$  on  $VM_j$ ,  $FT(T_i, VM_j) = ST(T_i, VM_j) + ET(T_i, VM_j)$ , where  $ST(T_i, VM_j)$  is the start time of task  $i$  on  $VM_j$  and  $ET(T_i, VM_j)$  is the execution time of task  $i$  on  $VM_j$ . Finally, the algorithm checks if it is equal to or less than the deadline constraint or not. If the finish time equals to or less than the deadline constraint, the algorithm assigns  $T_i$  to the selected  $VM_j$ . Otherwise, the algorithm searches for another suitable VM (see steps 8-20). Then, the algorithm moves to assign the tasks of the next level until it finishes all levels.

## C. THE EIDA ALGORITHM

For better balancing of the workload, the EIDA algorithm provides improved scheduling considering level load balancing (LLB). The algorithm tries to distribute the load in levels so that the children nodes of that level could not cause the delay in execution and also can balance the execution time of tasks of each level. The Child ascension process is one of the ways to achieve a level of load balancing. In this process, children of smaller tasks termed as the best suitable eligible children of that level and they must be ascended before their parents and executed in the same instance as their parents. This would reduce the execution delay created by the children nodes and also minimize the communication cost between children and their parents.

The immediate predecessors of task  $t_j$  are defined as:

$$iPred(t_j) = \{t_i | e_{ij} \in E\}, \quad (12)$$

where task  $t_i$  is a parent of task  $t_j$ .

Similarly, the successors of task  $t_i$  are defined as:

$$iSucc(t_i) = \{t_j | e_{ij} \in E\}, \quad (13)$$

where task  $t_j$  is a child of task  $t_i$ .

**Algorithm 3** The EIDA Algorithm**Input:** A workflow**Output:** A load-balanced schedule

1. Divide the workflow into levels;
2. Sort the levels according to the dependency order;
3. Take a workflow sample of level  $l$ ;
4. **For** all tasks in level  $l$  **do**
5.  $T_m =$  the biggest task in the level  $l$ ;
6. **For** each task  $t \in \{(TSLN(l) - T_m)\}$  **do**
7.  $t_{temp} = t$ ;
8.  $sum = \xi_t$ ;
9. **while** true **do**
10.  $ec = \emptyset$ ; // the set of eligible children
11. **For** each child task in  $l$  **do**
12. **If** the child task has no parents outside  $l$  and child task completion  $<$  the completion time of  $T_m$  **then**
13. Add child task to  $ec$ ;
14. **EndIF**
15. **EndFor**
16. **If**  $ec$  is empty **then**
17. **Break**;
18. **Else**
19. **If** predecessor exists **then**
20.  $bestchild = \text{Min}\{Rank_d(t_i)\}$ ;
21.  $sum = sum + bestchild$ ;
22.  $t_{temp} = bestchild$ ;
23. check for the next task;
24. **EndIF**
25. **EndIF**
26. **EndWhile**
27. **EndFor**
28. **EndFor**

The task without parents is termed as the entry task and the task without children is termed as the exit task.

$$iPred(t_{entry}) = \{\text{NULL}\}. \quad (14)$$

$$iSucc(t_{exit}) = \{\text{NULL}\}. \quad (15)$$

The finishing time of workflow is called the schedule length or makespan and it can be calculated by finding the finishing time of the exit task  $t_{exit}$ , and it is termed as  $L_{ms}$ :

$$L_{ms} = E(t_{exit}). \quad (16)$$

Communication time is defined as the amount of data transferred from the task  $t_i$  to task  $t_j$  and it is termed as  $C_{i,j}$ :

$$C_{i,j} = \begin{cases} \frac{D}{\beta}, & p_i \neq p_j \\ 0, & p_i = p_j. \end{cases} \quad (17)$$

where  $D$  is the amount of data transfer between the task  $t_i$  and the task  $t_j$  while  $\beta$  is the average bandwidth.

If the task  $t_i$  and task  $t_j$  are executed on the same processor instance, then communication cost is zero because no

communication happened between these tasks and if both the tasks are executed on different instances then the communication cost is the ratio of the total amount of data transfer between the task  $t_i$  and  $t_j$  and average bandwidth.

The completion time of  $t_i$  is calculated by adding the execution time and max communication time of the next level of the task and it is defined as:

$$\xi_{t_i} = E(t_i) + \max_{t_j \in iSucc(t_i)} \{C_{i,j}\}. \quad (18)$$

The level of the task  $t_i$  is the maximum number of edges from the task  $t_i$  to the exit task and it is an integer number. For the exit task, the level number is set to 1 and we move in the DAG from the exit task upward to the entry task. The level of task  $t_i$  is calculated by:

$$L(t_i) = \max_{t_j \in iSucc(t_i)} \{L(t_j) + 1\}, \quad (19)$$

where  $iSucc(t_i)$  represents the immediate successors of the task  $t_i$ .

Based on the level number, all the tasks at the same level are grouped and formed the Task Level Set Number (TSLN):

$$TSLN(l) = \{t_i | L(t_i) = l\}, \quad (20)$$

where  $l$  is the level number ranges from  $\{1, \dots, L(t_{entry})\}$ .

The downward rank  $rank_d(t_i)$  of a task  $t_i$  is computed recursively by traversing the direct acyclic graph from the entry task to the task  $t_i$ .

$$rank_d(t_i) = \max_{t_j \in iPred(t_i)} (w_j + C_{j,i} + rank_d(t_j)), \quad (21)$$

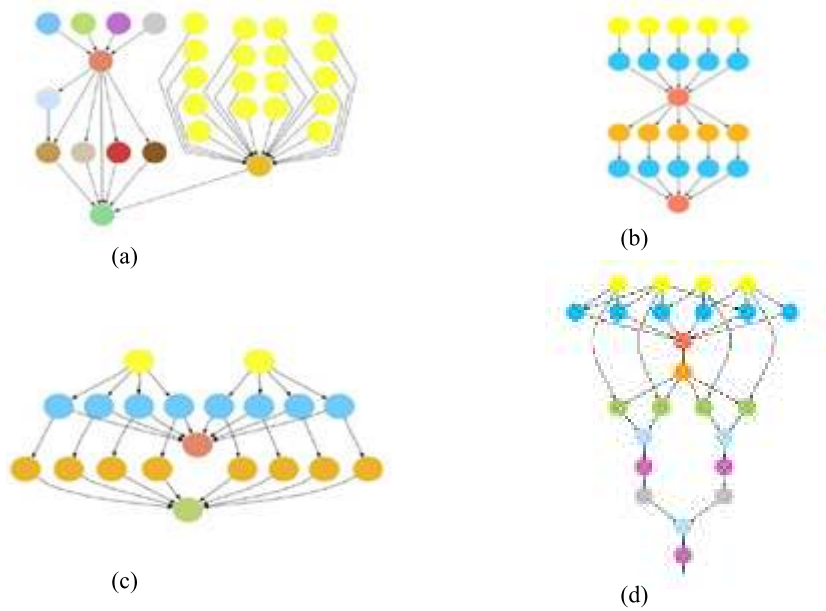
where  $w_j$  is the average computation cost of task  $t_j$ .

## V. RESULTS AND DISCUSSIONS

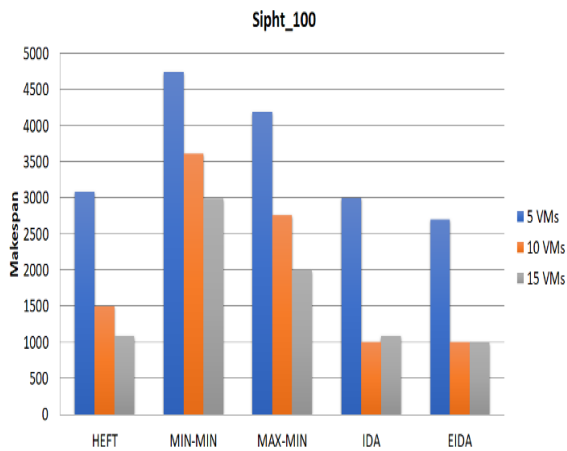
The main idea of the proposed system is to execute the submitted workflows with meeting the deadline and minimizing the cost. Besides, the new system applies a load balance phase to improve the utilization of the cloud computing system. The load balance phase tries to increase the utilization of the system without increasing the cost or the makespan.

### A. WORKFLOW MODEL

For evaluating the performance of the cloud-based data-intensive workflow scheduling, different scientific workflow applications are considered. Figure 5 shows some examples of scientific workflows of a small number of tasks such as Montage workflow, CyberShake workflow, Sipt workflow and LIGO workflow [30]. Montage [31] is an astronomical application created by NASA/IPAC for the stitches together multiple input images to the creation of custom mosaics of the sky generated by the. CyberShake [32] is used in earthquake science for the classification of earthquake hazards in a region and the Southern California Earthquake center uses it and it is a data-intensive application with very large memory and computing power requirement. Sipt [33] used in NCBI database for the automation of the searching of untranslated RNAs in bacterial replicons. LIGO [34] used in gravitational science for the detection of waveforms generated by the



**FIGURE 5. Examples of Scientific workflows (a) Sipt workflow (b) Inspiral Workflow (c) CyberShake workflow (d) Montage workflow.**



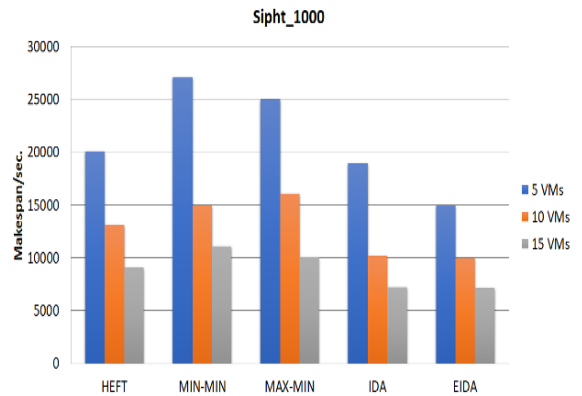
**FIGURE 6. Makespan of Sipt\_100 workflows for 5, 10, and 15 VMs.**

various events in the universe. This workflow is characterized by a large number of intensive tasks require a high amount of memory and computing power.

**B. SIMULATION RESULTS**

We have used the workflowsim tool [35] to develop the proposed system and evaluate it through comparison with the Min-Min, Max-Min, and the HEFT algorithms as the famous scheduling algorithms in cloud computing.

The experiments in this work are implemented using workflows from different organizations. Some workflows contain 100 tasks and others have 1000 tasks. The four types of the workflows, mentioned above, are employed in the experiments. The major factors used for comparison are makespan and cost.



**FIGURE 7. Makespan of Sipt\_1000 workflows for 5, 10, and 15 VMs.**

**• Makespan**

Makespan is the maximum time of the VM among virtual machines used to execute the workflow as shown in equation (22):

$$\text{Makespan} = \text{Max}(VM_{time}), \tag{22}$$

where  $VM_{time}$  is the finish time of VM. The makespan is considered a very important factor in evaluating the proposed management system. We apply the proposed system before and after the load balance phase. Before the load balance phase, the system is called IDA and after the load balance phase, we call it Enhanced IDA (EIDA).

As shown in Figures 6-13, the proposed system is more efficient than the HEFT, the Min-Min and the Max-min algorithms for most workflow types in terms of makespan. Moreover, the makespan is always less than the deadline required.



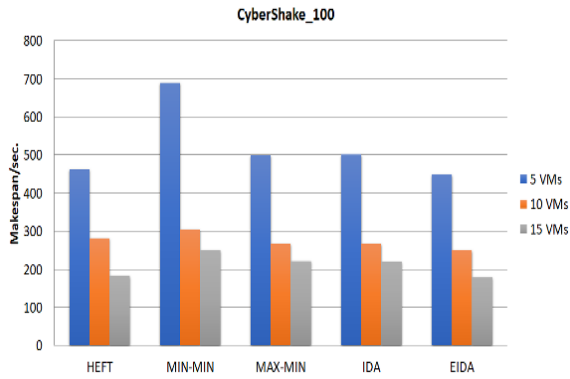


FIGURE 8. Makespan of CyberShake\_100 workflows for 5, 10, and 15 VMs.

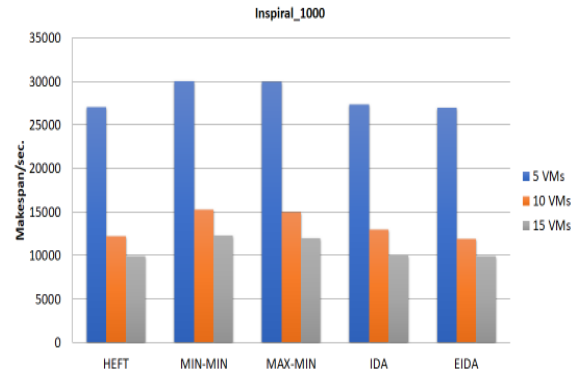


FIGURE 11. Makespan of Inspirial\_1000 workflows for 5, 10, and 15 VMs.

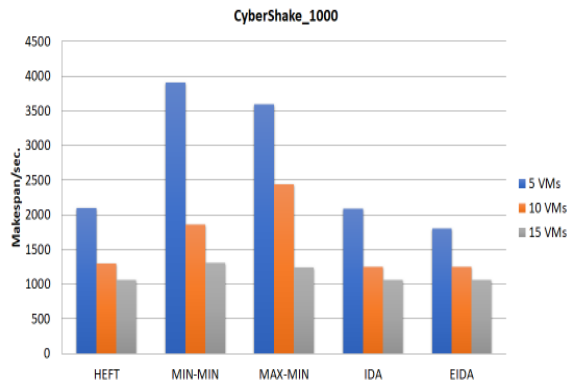


FIGURE 9. Makespan of CyberShake\_1000 workflows for 5, 10, and 15 VMs.



FIGURE 12. Makespan of Montage\_100 workflows for 5, 10, and 15 VMs.

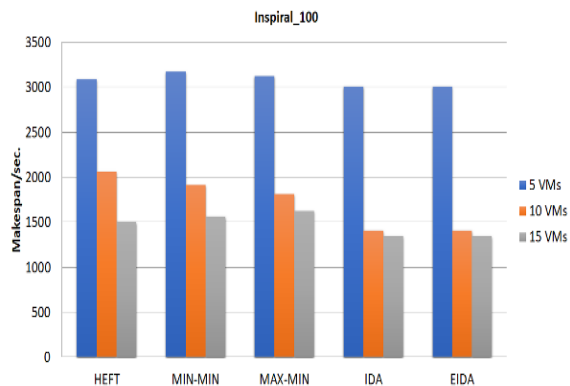


FIGURE 10. Makespan of Inspirial\_100 workflows for 5, 10, and 15 VMs.



FIGURE 13. Makespan of Montage\_1000 workflows for 5, 10, and 15 VMs.

However, in the Montage workflow case, the proposed system gives makespan higher than the other algorithms.

By applying the IDA with the load balance phase (EIDA), the makespan could be decreased. This appears for all cases, where the makespan of the EIDA algorithm is lower than the makespan of both the IDA and the HEFT algorithms for all workflow types.

• Cost

We select the Amazon EC2 for our pricing model scheme because of its widespread application. Amazon EC2

TABLE 1. VM pricing model.

Instance Type	vCPU	Compute unit	Memory ( GB)	Instance Storage(GB)	Price/Hour
t3.small	2	Variable	2	EBS only	\$0.0208
t3.large	2	Variable	8	EBS only	\$0.0832
t2.large	2	Variable	8	EBS only	\$0.0928
m5.large	2	8	8	EBS only	\$0.096
m5.xlarge	4	16	16	EBS only	\$0.0192

provides various instances to its end-users such as on-demand, reserved instances, spot instances and dedicated

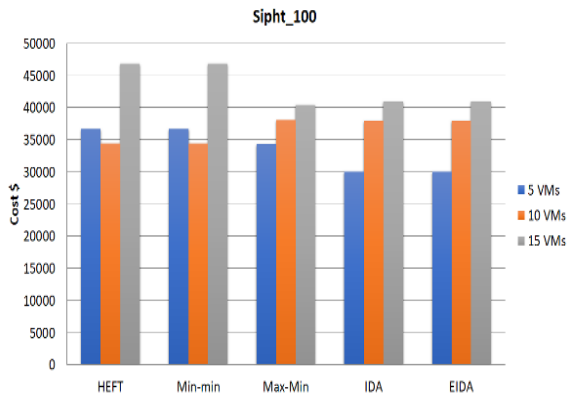


FIGURE 14. Cost of Sipht\_100 workflows for 5, 10, and 15 VMs.

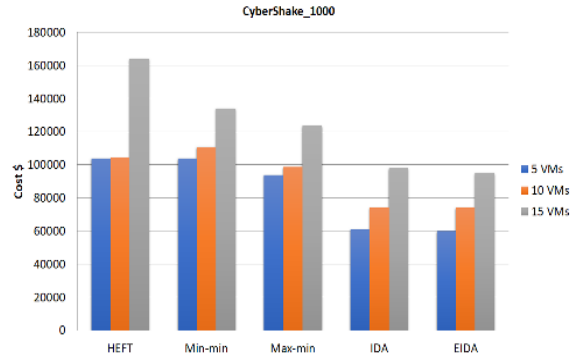


FIGURE 17. Cost of CyberShake\_1000 workflows for 5, 10, and 15 VMs.

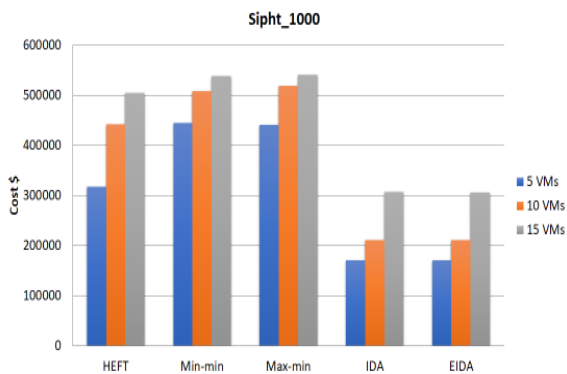


FIGURE 15. Cost of Sipht\_1000 workflows for 5, 10, and 15 VMs.

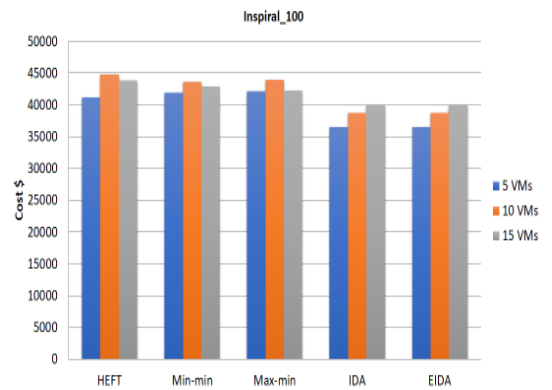


FIGURE 18. Cost of Inspiral\_100 workflows for 5, 10, and 15 VMs.

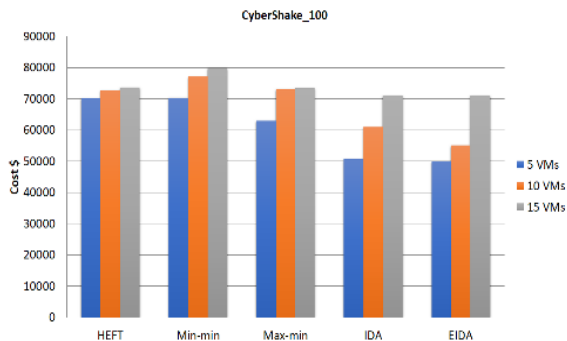


FIGURE 16. Cost of CyberShake\_100 workflows for 5, 10, and 15 VMs.

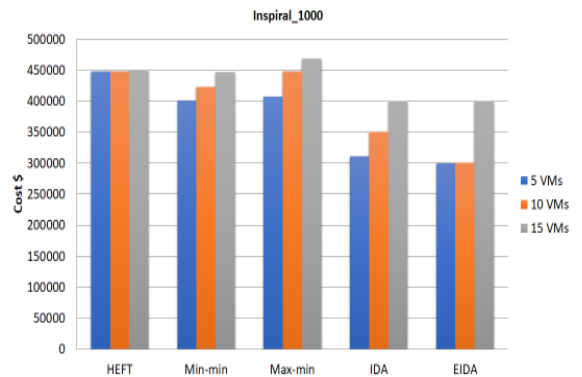


FIGURE 19. Cost of Inspiral\_1000 workflows for 5, 10, and 15 VMs.

hosts [29]. On-demand instances are the most used general-purpose instances with purchasing options in the US east regions. Table 1 shows the relevant parameter like instance type, CPU required, memory, bandwidth, storage type and price. In our experiment, we only take instances provided by US east regions since using Amazon Web Services (AWS) to run workflow needs to be on one specific region as shown in Table 1.

The cost is the price that users pay to get the service. In this paper, we are interested in the cost as the time. However, all results meet the deadline constraint. As shown

in Figures 14–21, it is clear that the results of the IDA are more efficient than the other algorithms. It can give inexpensive solutions.

In summary, the IDA algorithm is very useful for cloud vendors who are interested in presenting low-cost services and meet the deadline. It is not suitable for vendors who are considering only the time. We can use the EIDA algorithm in the case we want to tradeoff between the cost and the time. The results of the EIDA, in the case of the cost, are close to the results of the IDA algorithm.

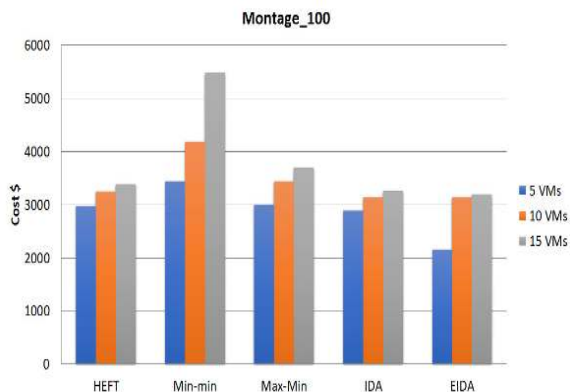


FIGURE 20. Cost of Montage\_100 workflows for 5, 10, and 15 VMs.

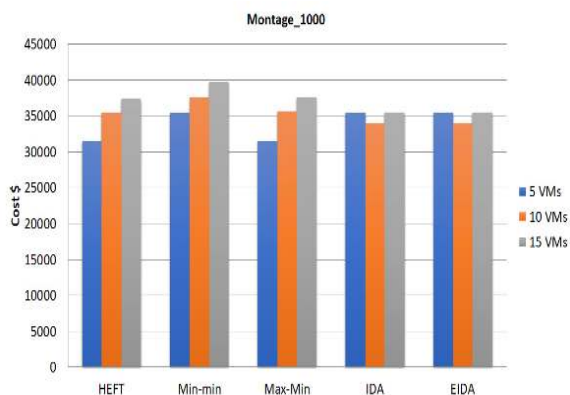


FIGURE 21. Cost of Montage\_1000 workflows for 5, 10, and 15 VMs.

## VI. CONCLUSION AND FUTURE WORK

Generally, a community cloud management system among organizations is an important matter to enhance system performance. The Workflow scheduling problem in the community model is the most challenging problem cloud computing model. Numerous studies conducted about the workflow scheduling and found that many heuristic and meta-heuristic algorithms provide the appropriate solution. In this paper, we proposed a new management system for servicing multi-organizations in a community cloud model. The system consists of three algorithms: Algorithm 1 for resource allocation, Ideal Distribution Approach (IDA) and Enhanced Ideal Distribution Approach (EIDA) heuristic-based algorithms are proposed to find the feasible and optimal schedule for the workflow execution to minimize makespan and cost while considering the deadline. IDA algorithm is very efficient for saving time and meeting the SLA terms. From the results, we can observe that the new system can organize the work between the different organizations. In addition, the proposed IDA achieves better solutions and also reduces the computational cost. The efficiency of IDA algorithm is enhanced by adding a load balance phase. The Enhanced IDA (EIDA) attains lower makespan as compared to IDA approach. From the experiment results, we can find that the EIDA outperforms

other conventional algorithms- HEFT, Min-Min and Max-Min in terms of makespan and cost. In addition to this, the proposed approach reduces the time complexity.

Future work in pipeline with considering work on full workflow application to find out the variation on cost, makespan and other parameters in the real cloud environment. Further security services for the cloud service provider also consider.

## REFERENCES

- [1] D. Owens, "Securing elasticity in the cloud," *Commun. ACM*, vol. 53, no. 6, pp. 46–51, 2010.
- [2] A. A. Nasr, A. T. Chronopoulos, N. A. El-Bahnasawy, G. Attiya, and A. El-Sayed "A novel water pressure change optimization technique for solving scheduling problem in cloud computing," *Cluster Comput.*, vol. 22, no. 2, pp. 601–617, 2019.
- [3] 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.
- [4] J. Meena, M. Kumar, and M. Vardhan, "Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint," *IEEE Access*, vol. 4, pp. 5065–5082, 2016.
- [5] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance analysis of high performance computing applications on the Amazon Web services cloud," in *Proc. 2nd IEEE Int. Conf. Cloud Comput. Technol. Sci.*, Nov./Dec. 2010, pp. 159–168.
- [6] Q. Wu, F. Ishikawa, Q. Zhu, Y. Xia, and J. Wen, "Deadline-constrained cost optimization approaches for workflow scheduling in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3401–3412, Dec. 2017.
- [7] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," *Future Gener. Comput. Syst.*, vol. 25, no. 5, pp. 528–540, 2009.
- [8] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generat. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.
- [9] M. Amoon, N. El-Bahnasawy, and M. ElKazaz, "An efficient cost-based algorithm for scheduling workflow tasks in cloud computing systems," *Neural Comput. Appl.*, vol. 31, no. 5, pp. 1353–1363, May 2019.
- [10] W. Zheng, B. Emmanuel, and C. Wang, "A randomized heuristic for stochastic workflow scheduling on heterogeneous systems," in *Proc. 3rd Int. Conf. Adv. Cloud Big Data*, Oct./Nov. 2015, pp. 88–95.
- [11] D. Sun, G. Zhang, S. Yang, W. Zheng, S. U. Khan, and K. Li, "Re-Stream: Real-time and energy-efficient resource scheduling in big data stream computing environments," *Inf. Sci.*, vol. 319, pp. 92–112, Oct. 2015.
- [12] 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.
- [13] I. Pietri and R. Sakellariou, "Cost-efficient CPU provisioning for scientific workflows on clouds," in *Proc. Int. Conf. Econ. Grids, Clouds, Syst., Services*. Cham, Switzerland: Springer, 2015, pp. 49–64.
- [14] S. Su, J. Li, Q. Huang, X. Huang, K. Shuang, and J. Wang, "Cost-efficient task scheduling for executing large programs in the cloud," *Parallel Comput.*, vol. 39, nos. 4–5, pp. 177–188, 2013.
- [15] T. A. L. Genez, I. Pietri, R. Sakellariou, L. F. Bittencourt, and E. R. M. Madeira, "A particle swarm optimization approach for workflow scheduling on cloud resources priced by CPU frequency," in *Proc. IEEE/ACM 8th Int. Conf. Utility Cloud Comput. (UCC)*, Dec. 2015, pp. 237–241.
- [16] Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1344–1357, May 2016.
- [17] R. Sakellariou, H. Zhao, E. Tsiakkouri, and M. D. Dikaiakos, "Scheduling workflows with budget constraints," in *Integrated Research in GRID Computing*, S. Gortlach and M. Danelutto, Eds. Boston, MA, USA: Springer, 2007, pp. 189–202.
- [18] K. Dubey, M. Kumar, and S. C. Sharma, "Modified HEFT algorithm for task scheduling in cloud environment," *Procedia Comput. Sci.*, vol. 125, pp. 725–732, 2018.

- [19] W. Zheng, Y. Qin, E. Buggingo, D. Zhang, and J. Chen, "Cost optimization for deadline-aware scheduling of big-data processing jobs on clouds," *Future Gener. Comput. Syst.*, vol. 82, pp. 244–255, May 2018.
- [20] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, vol. 49, Nov. 2011, pp. 1–12.
- [21] N. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2012, pp. 1–11.
- [22] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 158–169, Jan. 2013.
- [23] D. Poola, S. K. Garg, R. Buyya, Y. Yang, and K. Ramamohanarao, "Robust scheduling of scientific workflows with deadline and budget constraints in clouds," in *Proc. Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, May 2014, pp. 858–865.
- [24] E.-K. Byun, Y.-S. Kee, J.-S. Kim, and S. Maeng, "Cost optimized provisioning of elastic resources for application workflows," *Future Gener. Comput. Syst.*, vol. 27, no. 8, pp. 1011–1026, Oct. 2011.
- [25] S. C. Nayak, "Multicriteria decision-making techniques for avoiding similar task scheduling conflict in cloud computing," *Int. J. Commun. Syst.*, to be published, doi: [10.1002/dac.4126](https://doi.org/10.1002/dac.4126).
- [26] S. Nayak, S. Parida, and P. Pattnaik, "An enhanced deadline constraint based task scheduling mechanism for cloud environment," *J. King Saud Univ. Comput. Inf. Sci.*, to be published, doi: [10.1016/j.jksuci.2018.10.009](https://doi.org/10.1016/j.jksuci.2018.10.009).
- [27] F. Hao, G. Min, J. Chen, F. Wang, M. Lin, C. Luo, and L. T. Yang, "An optimized computational model for multi-community-cloud social collaboration," *IEEE Trans. Services Comput.*, vol. 7, no. 3, pp. 346–358, Jul./Sep. 2014, doi: [10.1109/TSC.2014.2304728](https://doi.org/10.1109/TSC.2014.2304728).
- [28] A. M. Khan, F. Freitag, and L. Rodrigues, "Current trends and future directions in community edge clouds," in *Proc. IEEE 4th Int. Conf. Cloud Netw. (CloudNet)*, Niagara Falls, ON, Canada, Oct. 2015, pp. 239–241.
- [29] G. Singh and M. Garg, "Data security in cloud computing: A review," *Int. J. Comput. Technol.*, vol. 17, no. 2, pp. 7206–7214, 2018.
- [30] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *Proc. 3rd Workshop Workflows Support Large-Scale Sci. (WORKS)*, Nov. 2008, pp. 1–10.
- [31] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, and M.-H. Su, "Montage: A grid-enabled engine for delivering custom science-grade mosaics on demand," in *Proc. 16th Annu. Symp. Electron. Imag. Sci. Technol. (IS&T/SPIE)*, 2004, pp. 221–232.
- [32] E. Deelman, S. Callaghan, E. Field, H. Francoeur, R. Graves, N. Gupta, V. Gupta, T. H. Jordan, C. Kesselman, P. Maechling, J. Mehlinger, G. Mehta, D. Okaya, K. Vahi, and L. Zhao, "Managing large-scale workflow execution from resource provisioning to provenance tracking: The cyberShake example," in *Proc. 2nd IEEE Int. Conf. e-Sci. Grid Comput. (e-Sci.)*, Dec. 2006, pp. 4–6.
- [33] *Workflow Generator*. Accessed: Jun. 15, 2019. [Online]. Available: <https://confluence.pegasus.isi.edu/display/Pegasus/WorkflowGenerator>
- [34] H. M. Monti, A. R. Butt, and S. Vazhkudai, "CATCH: A cloud-based adaptive data transfer service for HPC," in *Proc. 25th IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2011, pp. 1242–1253.
- [35] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. IEEE 8th Int. Conf. E-Sci.*, Chicago, IL, USA, Oct. 2012, pp. 1–8.



KALKKA DUBEY received the B.E. degree in computer science and engineering from MITS Gwalior Autonomous, in 2010, and the M. Tech. degree in computer science and engineering from ABV-IIITM Gwalior, in 2013. He is currently pursuing the Ph.D. degree with IIT Roorkee, India. His research interests include practical and experimental and are focused on task scheduling and VM placement and allocation in cloud-based systems, quantification and monitoring of security metrics, and enforcing security in cloud environments.

**KALKKA DUBEY** received the B.E. degree in computer science and engineering from MITS Gwalior Autonomous, in 2010, and the M. Tech. degree in computer science and engineering from ABV-IIITM Gwalior, in 2013. He is currently pursuing the Ph.D. degree with IIT Roorkee, India. His research interests include practical and experimental and are focused on task scheduling and VM placement and allocation in cloud-based systems, quantification and monitoring of security metrics, and enforcing security in cloud environments.



MAHMOUD Y. SHAMS received the bachelor's degree in electronics and communication from the Faculty of Engineering, Mansoura University, in 2004, the master's degree in computer vision and pattern recognition from the Faculty of Computer and Information Sciences, Mansoura University, and the Ph.D. degree from the Computer Science Department, Mansoura University. He is currently an Assistant Professor with the Machine Learning and Information Retrieval Department, Faculty of Artificial Intelligence, Kafr Elsheikh University. He has published over ten articles in refereed international journals. His research interests include using deep learning approaches and cloud computing.



S. C. SHARMA received the M.Sc. degree in electronics, the M.Tech. degree in electronics and communication engineering and the Ph.D. degree in electronics and computer engineering from IIT Roorkee. He is currently a Professor with IIT Roorkee. He has more than 25-year Research and Teaching Experience. He has published over 200 research articles in national and international journals/conferences and supervised 16 Ph.D. students. He received the Khosla Research Prize for the Best Research Paper from IIT Roorkee. He was a Research Scientist with FMH, Germany.



ABDULAZIZ ALARIFI received the Ph.D. degrees in information security from the University of Wollongong, Australia. He is currently an Assistant Professor with the Computer Science Department, Community College, King Saud University (KSU), Saudi Arabia, where he is also the Head of research unit with the Community College. His main research interests include information security, information technology management, cloud computing, big data, information privacy, risk assessment and management, e-governance, and mobile applications.



MOHAMMED AMOON received the B.Sc. degree in electronic engineering and the M.Sc. and Ph.D. degrees in computer science and engineering from Menoufia University, in 1996, 2001, and 2006, respectively. He is currently a Professor with the Computer Science and Engineering Department, Menoufia University and the Computer Science Department, King Saud University. His research interests include agent-based systems, fault tolerance, green computing, distributed computing, grid computing, cloud computing, and fog computing.



**AIDA A. NASR** received the B.Sc. and M.Sc. degrees from the Faculty of Electronic Engineering, Menoufia University, Egypt, and the Ph.D. from the Department of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufia University, Egypt, in 2019. She is currently an Assistant Professor with the Faculty of Artificial Intelligence, Kafrelsheikh University, Egypt. Her research interests include agent-based systems, distributed computing, grid computing, cloud computing, and fog computing.

...