CHAPTER 3

MEETING DEADLINES USING ARTIFICIAL BEE COLONY (ABC) BASED RESOURCE MECHANISM IN PUBLIC CLOUDS WITH TASKS REPLICATION

3.1 INTRODUCTION

Scientific workflows provide a group of features, which create them appropriate for execution in cloud infrastructures that present on-demand scalability, which lets resources to be raised and reduced with the intention of acclimatizing to the request of applications Foster et al. (2008). Previous research in implementation of scientific workflows in clouds attempts to reduce the workflow execution time disregarding targets and budgets or concentrates on the reduction of cost when aiming to reach the application target and also does not reach the identification of the execution time of every task in the workflow.

With the aim of stating disadvantages of existing research, an Enhanced Artificial Bee Colony (ABC) based IaaS Cloud Partial Critical Path (IC-PCP) with Replication algorithm known as EAIPR is proposed. It improves the probability of finishing the implementation of a scientific workflow application within a user-defined deadline in a public cloud environment. ABC algorithm is utilized to identify the parameters of tasks. They are the early start time (est) as well as the latest finish time (lft) thatimitate workflow tasks to reduce consequences of performance deviation of resources with the intension that soft deadlines could be attained. The
EAIPR algorithm utilizes idle time of provisioned resources to imitate workflow tasks in order to alleviate consequences of performance deviation of resources with the intension that soft deadlines could be attained.

### 3.1.1 Importance of Resource Management and PCP

A cloud computing infrastructure is a composite system containing huge amount of shared resources. They depend on random requests and could be influenced by external events outside control. Cloud resource management needs composite policies as well as choices for multi-objective optimization. It is very much challenging, due to the complexity of the system that creates it difficult to contain precise global state information. It depends on continual as well as random interactions with the environment. Resource management is a main function needed for any artificial system. It has an impact on three fundamental criteria for system assessment and they are functionality, performance, and cost. Ineffective resource management shows a straight deleterious impact on cost as well as performance. It ramblingly has an impact on the functionality of the system. Certain functions of the system may turn out to be very costly or unsuccessful because of reduced performance.

Numerous researchers have presented several algorithms for allotting, scheduling as well as scaling the resources professionally in the cloud. They comprises min-min ,max min, first come first serve, ant colony, earliest dead line first, round robin, back tracking, hybrid heuristic, task duplication, loss and gain simulated annealing, genetic algorithm, greedy, ant colony and so on as explained by Jayarani et al.(2009). Numerous modified scheduling algorithms such as Modified Ant Colony Optimization, Improved Genetic Algorithm, Extended Min-Min have also been developed (Calheiros et al. 2009 & Singh et al. 2010).
The arrival of cloud computing as a novel exemplary of service provisioning in distributed systems inspires researchers to examine its advantages as well as disadvantages on implementing scientific applications for instance workflows. The most difficult problem in clouds is workflow scheduling and that is to say the issue of fulfilling the QoS requirements of the user and reducing the workflow execution cost. Numerous systems have been developed as well as examined a two-phase scheduling algorithm for utility Grids, known as Partial Critical Paths (PCP) that focuses on reducing the workflow execution cost, when attaining a user defined deadline. On the other hand, there are three important dissimilarities amidst the present commercial clouds as well as the utility grid model for which, the PCP algorithm has been developed. The primary dissimilarity is the on-demand (dynamic) resource provisioning feature of the clouds and that facilitates the scheduling algorithm for identifying the type and the number of needed resources in utility grids. There are pre-set and limited numbers of resources with limited existing time slots. This property provides the impression of unrestrained resources to the cloud users. The next difference is the (almost) homogeneous bandwidth amidst the services of a cloud provider, in competition with the heterogeneous bandwidth amidst the service providers in the utility grids. The final (and most significant) dissimilarity is the pay-as-you-go pricing model of present commercial clouds in which the users will be charged depending upon the amount of the time intervals they have utilized. Since the time interval is long (for instance one hour in Amazon EC2) and the user is totally charged for the last time interval even if the user utilizes just a small portion of it, the scheduling algorithm must attempt to use the last interval as greatly as possible. By taking these dissimilarities, the PCP algorithm is acclimatized and two new workflow scheduling algorithms are presented for IaaS clouds. They are known as the IaaS Cloud-Partial Critical Path (IC-PCP).
3.1.2 Introduction about Artificial Bee Colony (ABC)

According to ABC algorithm, the colony of artificial bees encompasses three types of bees: employed bees, onlookers and scouts. In this algorithm, initial half of the colony encompasses employed bees and the second half comprises the onlookers. There exists just one employed bee for each food source. Alternatively, the amount of employed bees is equivalent to the amount of food sources around the hive described by Gao et al. (2013). The employed bee, whose food source is abandoned by the bees, turns out to be a scout as explained in (Karaboga et al. 2014).

The onlookers notice the dance of the employed bees inside the hive, for choosing a food source, while scouts look for arbitrarily for novel food sources. Analogously in the optimization perspective, the amount of food sources (i.e., the employed or onlooker bees) in ABC algorithm is equal to the amount of solutions in the population. Additionally, the location of a food source indicates the location of a hopeful solution to the optimization issue, while the nectar quality of a food source signifies the fitness cost (quality) of the related solution explained in (Nocedal & Wright 2006).

The search cycle of ABC encompasses three rules: (i) directing the employed bees to a food source and assessing the nectar quality (ii) onlookers selecting the food sources subsequently getting information from the employed bees and computing the quality of nectar (iii) identifying the scout bees and directing them to probable food sources as explained in (Karaboga et al. 2008). The locations of the food sources are arbitrarily chosen by the bees at the primary phase and their nectar qualities are deliberated and explained in (Suguna & Thanushkodi 2011). After that, the employed bees will share the nectar information of the sources with the bees in the dance area inside the hive. Subsequently, each employed bee comes back to the food source visited during the period of the previous cycle, as the location of the food source has
been remembered and after that, chooses additional food source by means of utilizing its visual info in the neighbourhood of the current one. In the final phase, an onlooker utilizes the information got from the employed bees in the dance area for choosing a food source explained in (Gao et al. 2013). The likelihood of the food sources to be chosen rises with rise in its nectar quality. So, the employed bee with information of a food source with the maximum nectar quality employ the onlookers to that source explained in (Akay & Karaboga 2012). Then, it selects one more food source in the neighbourhood of the one presently in its remembrance depending upon its visual information (that is to say evaluation of food source locations). A novel food source is arbitrarily produced by a scout bee to swap the one abandoned by the onlooker bees as explained in (Li & Jian 2009). This search process is denoted in Algorithm (1)

**Algorithm 1:** Psudeocode of ABC procedure

Step 1: Initialize the food source positions  
Step 2: Assess the food sources  
Step 3: Produce new food sources (solutions) for the employed bees  
Step 4: Employ greedy selection  
Step 5: Compute the fitness and probability values  
Step 6: Produce new food sources for onlookers  
Step 7: Employ greedy selection  
Step 8: Identify the food source to be abandoned and allot its employed bee as a scout for looking for the novel food sources  
Step 9: Memorize the best food source found
Step 10: Do again steps 3-9 for a pre-determined number of iteration

In this research, Resource constraint management is carried out by means of ABC algorithm.

3.2 PROPOSED METHODOLOGY

In order to minimize the total execution cost of a workflow, first research work introduces an Enhanced Artificial Bee Colony (ABC) based IaaS Cloud Partial Critical Path (IC-PCP) with Replication algorithm is called as EAIPR. In the IC-PCP algorithm, the estimation of the time to complete the data transfer time for each task in the VM becomes difficult, if the number of tasks becomes high. In the resource provisioning problem, the objective function such as the early start time (est) and latest finish time (lft) are calculated using the Artificial Bee Colony (ABC) algorithm. The goal of the proposed EAIPR algorithm is increasing the likelihood of completing the execution of a scientific workflow application within a user-defined deadline in a public cloud environment. This typically offers high availability but gives significant performance variation with the use of task replication.

3.2.1 System Model

Workflow applications are composed of dependent tasks and modelled as Directed Acyclic Graphs (DAGs), where each task can only start its execution, after all its predecessors have finished and the data have been transferred to the machine where it is scheduled to execute. Each workflow task (i.e., each node of the DAG) has a computational demand associated and it is translated into how long it takes to run according to the CPU capacity of each VM. Also, the data transmission between two tasks occurs in the network that connects the VMs where these two tasks are scheduled. If they are scheduled to the same VM, the data transmission time between them is
zero. During the workflow execution, the necessary data to each task can be kept into the storage available in the VM where the task is running, or it can be stored in extra storage from rented storage facilities. While the VM-storage cost is included in the VM price, the extra storage is charged independently. Moreover, this extra storage is deployed over the network, and therefore, there exists a maximum transfer speed from this storage to the VM where the task is being run.

A scientific workflow application is modelled as a Direct Acyclic Graph (DAG) $G = (T, E_T)$, where $T$ is the set of tasks that consist of workflow and $E_T$ is the set of dependencies between tasks. Dependencies are in the form of edges $e_{i,j} = (t_i, t_j)$, $t_i, t_j \in T$ that establish a task $t_j$, depends on data generated by $t_i$ for its execution, and therefore, $t_j$ cannot start before the execution of $t_i$ completes and data generated by the latter is transferred to the location where $t_j$ will execute. Task $t_i$ is a parent task of $t_j$ and $t_j$ is a child task of $t_i$. Tasks without parents are called entry tasks and the tasks without children are called exit tasks. For the correct operation of the proposed algorithm, it is assumed that a workflow can have only one entry task and one exit task. This can be achieved with the insertion of “dummy” tasks $t_{entry}$ and $t_{exit}$ that have execution time equals to zero. All the actual entry tasks are children of $t_{entry}$ and all the actual exit tasks are parents of $t_{exit}$. The sets of parents and children of a task $t_j$ are given respectively by functions $\text{parents}(t_j)$ and $\text{children}(t_j)$. Each workflow $G$ has a soft deadline $dl(G)$ associated with it. It determines the time to complete its execution and it is counted from the moment that it is submitted to the workflow scheduler. The latter manages the execution of the workflow, makes decision on allocation of virtual machines, and schedules and dispatches tasks for execution in the cloud. Cloud provider offers a set of $n$ Virtual Machine (VM) types denoted by $\overline{VM} = \text{vm}_1, ..., \text{vm}_n$. Each VM type offers different amount of resources,
and incurs a different cost per use. Let \( \vec{c} = c_1, c_2, \ldots, c_n \) be the cost vector associated with the use of each VM. VMs are charged per integer amount of time units, and partial utilization of a time period incurs charge for the whole period. Therefore, if the time period is one hour, utilization of a VM per 61 minutes incurs in the payment of two hours (two time periods). There is no limit imposed on the number of VMs of each type that may run in any moment for the execution of the workflow.

Runtime of each task is defined in the runtimematrix \( R \). An element \( r_{jk} \) of \( R \) specifies the estimated runtime of task \( t_j \) in a VM of type \( vm_k \). The minimum runtime \( R_{\min}(t_i) \) of a task \( t_i \) is the smallest runtime for such a task in the matrix \( R \). Notice that \( r_{entryk} = r_{exitk} = 0 \) for all \( k \). Tasks cannot be pre-empted or check pointed. Therefore, if the execution of a task fails or if a task is cancelled by the scheduler, it has to be restarted. Each edge \( e_{i,j} \) of \( G \) has an associated data transfer time \( D(i,j) \). This is the amount of time required to transfer the data required by the non-entry and non-exit task \( t_j \) from the VM where \( t_i \) is running to the VM. Notice that, if both \( t_i \) and \( t_j \) are running on the same VM, \( D(i,j) = 0 \). The existence of data transfer time among different VMs implies that, for each task \( t_j \) to be executed in a given VM \( vm_k \), \( vm_k \) is deployed before the data transfer from parents of \( t_j \) start, and is decommissioned after all the data transferred to its children are completed.

Important parameters of tasks are the early start time (est) and latest finish time (lft). The former represents the earliest time a task is able to start, which happens, when all its parent tasks finish as early as possible. The latter represents the latest time a task can finish without missing the deadline, which happens, when all the children of a task are executed as late as possible. Formally, est and lft are defined in Equation (3.1) and Equation (3.2)
The schedule time $st(t_j)$ of a task $t_j$ is the time on which the task has been scheduled for execution. This parameter is defined during the scheduling process and it can assume any value between $est(t_j)$ and $lft(t_j)$.

### 3.2.2 Artificial Bee Colony (ABC) based Mechanism to Resource Management

The problem addressed in this research consists of the execution of a workflow $G$ in the cloud on or before $dl(G)$ (i.e., deadline-constrained) with a smaller possible cost (i.e., cost-optimized). Because, the workflows are subjected to a soft deadline, a bigger budget can be invested for execution of $G$, if it increases the likelihood of the deadline being met. The extra budget is expected to be proportional to the importance of the application to complete by its deadline. For this problem to be solved, two sub-problems have to be solved, namely provisioning and scheduling. The provisioning problem consists of the determination of the optimal number and type of VMs that can complete the workflow within its deadline. The scheduling problem comprises the determination of the placement and the order of execution of different tasks that compose the workflow of the VMs selected during the provisioning stage. The provisioning and scheduling problems are interconnected, as different decision types and number of machines may

\[
est(t_j) = \begin{cases} 0, & lft_t = \max_{t_{a} \in \text{parents}(t_j)} \left( est(t_a) + R_{\text{min}}(t_a) + D(e_{a,j}) \right) \\ \text{Exit}, & \text{Otherwise} \end{cases}
\]

\[
lft(t_j) = \begin{cases} dl(G), & lft_t = \max_{t_{s} \in \text{children}(t_j)} \left( lft(t_s) - R_{\text{min}}(t_s) + D(e_{j,s}) \right) \\ \text{Exit}, & \text{Otherwise} \end{cases}
\]
result in different scheduling of tasks. However, cloud environments do not present regular performance in terms of execution and data transfer time.

Jackson et al. (2010) have reported performance variation of up to 30 percent in execution time and up to 65 percent in data transfer time, when High Performance Computing (HPC) applications are executed in public clouds. This demands countermeasures to be applied at the application provisioning and scheduling stage to enable soft deadlines to be met. Therefore, the application of task replication is proposed as a means to mitigate the effect of performance variation of cloud resources in the workflow execution time. The IC-PCP algorithm is the state-of-the-art algorithm for provisioning and scheduling of workflows in clouds. The workflow and cloud VM model are proposed by Abrishami et al. (2013) as depicted in Figure 3.1. Tasks are represented as circles. The data transfer time between tasks, if running in different VMs, they are depicted in the arcs, and the execution times of tasks in three different VM types are labelled as S1, S2, and S3.

![Figure 3.1 Example of workflow and available VMs](image_url)
Numbers in arcs denote data transfer time, whereas the table presents execution time of tasks on three different VM types S1, S2, and S3, as well as the cost per time unit of each VM. Workflow’s deadline is 30 time units and allocation slot is 10 time units. The provisioning and scheduling problems are closely related, because the availability of VMs affects the scheduling, and the scheduling affects finish time of virtual VMs. Therefore, a more efficient scheduling and provisioning can be achieved, if both the problems are solved as one rather than independently.

Among the existing approaches of combined provisioned and scheduling of workflow applications in public cloud environments, the IC-PCP (IaaS Cloud Partial Critical Path) algorithm works with the closest assumptions of the system and application models. It is a cost-minimizer with deadline constraint algorithm that operates via assignment of all the tasks of a Partial Critical Path (PCP) of the workflow in the same Virtual Machine (VM). In the IC-PCP algorithm, the estimation of time to complete the data transfer time of each task in the VM becomes difficult, if the number of tasks becomes high. To solve this problem, Artificial Bee Colony (ABC) has been proposed in this work.

In the resource provisioning problem, the early start time (est) and latest finish time (lft) are calculated using the Artificial Bee Colony (ABC) algorithm as stated in Equations (3.1) and(3.2). The goal of the proposed Enhanced ABC based IC-PCP with Replication(EAIPR) algorithm is to increase the likelihood of completing the execution of a scientific workflow application within a user-defined deadline in a public cloud environment. It typically offers high availability but offers significant performance variations, with the use of task replication. In a high level, the proposed algorithm performs with four distinct steps.
- Combined provisioned of cloud resources
- Cloud resource management using the ABC and task scheduling
- Data transfer-aware provisioning adjust
- Task replication

The overall processes involved in the proposed research work is illustrated in the flowchart of Figure 3.2.

Figure 3.2 Architecture of the proposed system
3.2.3 Combined Provisioning and Scheduling

The first step of the EAIPR algorithm is to determine the number and type of VMs to be used for workflow execution as well as start and finish time of each VM (provisioning) and the determination of ordering and placement of tasks on such allocated resources (scheduling). The provisioning and scheduling problems are closely related because the availability of VMs affects the scheduling, and the scheduling affects the finish time of virtual VMs. Therefore, a more efficient scheduling and provisioning can be achieved, if both problems are solved as one rather than independently. Among the existing approaches of combined provisioned and scheduling of workflow applications in public cloud environments, the IC-PCP (IaaS Cloud Partial Critical Path) algorithm works with the closest assumptions of the system and application models.

It is a cost-minimizer with deadline constraint algorithm that operates via assignment of all the tasks of a Partial Critical Path (PCP) of the workflow in the same virtual machine. The PCP is initially composed of one of the parents of t_{exit} or, if it has already scheduled the task with the latest Lf that has not been assigned to a PCP. The next element to be part of the PCP is the parent t_p of such task with latest est(t_p)+R_{min}(t_p)+ D(e_p,o), i.e., the parent with longer execution and data transfer time. The procedure for definition of a PCP is detailed in Algorithm 2. Algorithm 2 is initially executed to t_{exit} and then, recursively for each task of the resulting PCP, starting from the first. The process is repeated for each parent of t_{exit}. The whole PCP is assigned to a machine, as detailed below, and est, Lf, and st of all tasks affected by the scheduling of the PCP are updated.

For assigning partial critical paths to VMs, the method is adopted from the IC-PCP algorithm. It examines the available VMs starting from the cheapest to the more expensive to find the one which is able to execute each
task of the path before its lft. The first VM, which is able to meet such requirement, is selected. If none of the existing machines can meet this requirement, a new VM is provisioned. This type of VM is the cheapest one that can execute each task of the critical path before the task’s lft. When evaluating the suitability of a VM for receiving a partial critical path, EIPR inserts the new path in either the beginning of the scheduling, if it does not violate lft or dependencies of previously assigned tasks (Step 5-11). Otherwise, it schedules the path at the end of the schedule (Step 14-18).

When evaluating each case, the case is considered invalid, if any of the following conditions occurs: 1) at least one scheduled task has its lft violated, if it is pre-empted 2) at least one task of the path has its lft violated, if the path is scheduled in the considered position and 3) it will require the VM to be executed for an extra time slot. The IC-PCP algorithm disregards deployment and boot time of virtual machines, by assuming that the earliest start time of the entry task is 0. However, virtual machines provisioned from public cloud providers are not immediately available for task execution. Hence VMs need to be properly initialized and this time is not negligible. To better model effects of such non-negligible deployment and boot times of virtual machines in the workflow scheduling process, the EIPR algorithm assigns the average boot time of virtual machines, rather than 0, to \( \text{est}(t_{\text{entry}}) \), and \( \text{st}(t_{\text{entry}}) \) before calculating \( \text{est} \) and \( \text{lft} \) of each task.

**Algorithm2: Assignment of the PCP of the task t**

Step 1: Data: \( U \): Unscheduled tasks of \( G \)

Step 2: \( \text{pcp} \leftarrow \emptyset \)

Step 3: while \( U \supseteq \text{parents}(t) \) do

Step 4: for each \( t_p \in \text{parents}(t) \) do
Step 5: if \( t_p \subseteq U \), then

Step 6: \( \text{readyTime} \leftarrow -1 \)

Step 7: estimate \( est, lft \) by ABC

Step 8: if \( est(t_p) + R_{\text{min}}(t_p) + D(t_p, t) > \text{readytime} \) then

Step 9: \( \text{readytime} \leftarrow est(t_p) + R_{\text{min}}(t_p) + D(t_p, t) \);

Step 10: \( \text{parent} \leftarrow est(t_p) \)

Step 11: end if

Step 12: end if

Step 13: end for

Step 14: \( \text{pcp} \rightarrow \text{parent} \cup \text{pcp}; \)

Step 15: remove parent from \( U \);

Step 16: \( t \leftarrow \text{parent}; \)

Step 17: end while

Step 18: return pcp

The foraging behaviour of honey bees has been adopted by the ABC for simulation. There happens a special dancing behaviour of the bees, during the search of early start time (est) and latest finish time (lft) for the execution of task in a workflow G in the cloud on or before dl (G). Each bee is considered as the number of tasks in a workflow G. The number of task in the employee bee in ABC optimization is used to visit the food source position and to find early start time (est) and latest finish time (lft). Each one of the employee bee in ABC gathers information about the task in the workflow G in the cloud with early start time (est) and latest finish time (lft).
Employed bees also perform the local investigation to find early start time (est) and latest finish time (lft). They try to exploit the exact early start time (est) and latest finish time (lft) of neighbouring results of each task in the workflow G. Those bees, that are waiting in the nest area to determine early start time (est) and latest finish time (lft) for each task in the workflow G, are termed as onlooker bees.

Onlooker bees perform the global investigation to determine early start time (est) and latest finish time (lft) of each task. They update global optimum early start time (est) and latest finish time (lft) thereby resulting in position update phase. The random discovery of new early start time (est) and latest finish time (lft) value of each task in the workflow G is not usually focused by the employed bees where it is performed by scout bees. These three steps are continued, until a maximum number of iteration termination criterion is satisfied. The fitness values of the Artificial Bee Colony (ABC) algorithm, to determine early start time (est) and latest finish time (lft) for each task in the workflow G, are shown in Algorithm 3. The position of the selected tasks is updated by using the following Equation (3.3)

\[ t_i = t_j + \theta_{ij}(t_{ij} - t_{id}) \]  

(3.3)

Where k and l are randomly selected tasks \(k \in \{1,2,\ldots,SN\}\) & \(j \in \{1,2,\ldots,D\}\). \(\theta_{ij} \in [-1,1]\). An artificial onlooker bee selects the tasks based on the calculation of the probability value \(p_a\) as stated in the Equation (3.4)

\[ p_a = \frac{f(t_i)}{\sum_{w=1}^{SN} f(t_j)} \]  

(3.4)

where \(f(t_j)\) represents the fitness value of each task (employee bee), i is the location and SN refers to the size of the population. In ABC, if a bee task value position does not improve the result within a pre-specified number of
iterations, then the current task to be assumed as neglected and it is updated in Equation (3.5)

\[ t_i^j = t_{\min}^j + \text{rand}(0,1)\left(t_{\max}^j - t_{\min}^j\right) \] (3.5)

All the above mentioned steps majorly depend on the following parameters which restrict the operation SN and the Maximum Number of the Cycles(MNC).

**Algorithm 3: Solve Resource provisioning problem in PCP using ABC**

**Input:** SN- size of bees is the number of tasks with execution time and D dimensions of search space

**Output:** Update est, lft, and st of all tasks affected by the scheduling of the PCP

- **Step1:** Initialize bees with number of tasks in the search space D
- **Step 2:** Set cycle = 1
- **Step 3:** Repeat
- **Step 4:** For all the tasks in the employee bee do
- **Step 5:** Evaluate the objective function of each task in Equation (3.1) and Equation (3.2)
- **Step 6:** Produce new est, lft, and st of all tasks by using Equation (3.3)
- **Step 7:** Apply the greedy selection process for the employed bees
- **Step 8:** Calculate the probability values \( p_a \) for selected task
- **Step 9:** Produce current task position solutions \( t_i^j \) for the
onlookers bees depending on $p_a$ and evaluate them

Step 10: Apply the greedy selection process for the onlookers

Step 11: Determine the abandoned results in the scout, if exists, replace it with a new randomly produced solution $t_i$ by using Equation (3.5)

Step 12: Memorize the best solution achieved so far

Step 13: $cycle = cycle + 1$

Step 14: until $cycle = MCN$ is met (usually sufficiently good weight value fitness)

Step 15: Return the best $est$ and $lft$ for each task, its fitness value

3.2.4 Data-Transfer Aware Provisioning Adjust:

The combined provisioning and scheduling detail in the previous section does not dictate the start and stop time of VMs. To determine both the values, the algorithm has to consider not only start and end time of the scheduled tasks, but also the data transfer to the first scheduled task and from the last scheduled task. If the first scheduled task in a VM is not an entry task, data from parent tasks have to be moved to the virtual machine before the task can run, and thus, the VM needs to be provisioned before the start time of its first task. This affects the start time of the tasks and the total provisioning time of the VM. Hence, it may cause workflow execution delay and execution of VMs for an extra billing period.

In the second step of the EIPR algorithm, the provisioning decision performed in Step 1 is adjusted to account for the aforementioned factors. For each non-entry task scheduled as first task of a virtual machine, and for each non-exit task scheduled as the last task of a virtual machine, the algorithm meets the required communication time by setting the start time of the
machine D(i,j) earlier than starting time of the first task, and/or setting the end time of the machine D (i,j), later than the finish time of the last task, depending on where the extra time is required. Finally, the beginning of the first allocation slot of each virtual machine is anticipated by the estimated deployment and the boot time for virtual machines, which are accounted during the scheduling process.

### 3.2.5 Task Replication

The aforementioned corrections enable virtual machines to be ready to receive data and tasks in the moment that they are required to meet time estimated during the scheduling process. However, it does not account for delays in the task executions caused by poor performance of public cloud resources. EIPR tries to mitigate such effects with the utilization of task replication in idle slots of provisioned VMs or on new VMs allocated for enabling extra replication (if the replication budget allows). The goal of this replication is to increase the performance rather than fault tolerance and the space replication is the target of EIPR. Therefore, tasks are only replicated on different VMs, oppositely to a time replication approach where the same task could be scheduled multiple times in a single VM to increase fault-tolerance.

Idle slots exist in the scheduling because of two reasons. The first is dependencies between tasks, which may lead to period where the next task scheduled to a virtual machine has to wait for data to be generated during the execution of another task in another machine. The other cause of idle slot, is due to the provisioning times of VMs. In some situations, some billing periods will be only partially used, and the excess can be used for task replication purposes. The task replication process works as a semi-active replication technique for fault tolerance, with the difference that here tasks are replicated across performance-independent hosts rather than failure-independent locations. As in the case of replication for fault tolerance, in the proposed approach, replication is also managed by a single entity.
3.3 PERFORMANCE EVALUATION

In this section, the experiments are conducted to evaluate the EIPR and EAIPR algorithms. The experiments have been conducted by using the CloudSim toolkit described in Juve et al. The simulation testbed consists of a data center containing 500 hosts. Each host has 256 GB of RAM and 8 cores. The data center models, Amazon AWS EC2 standard instance types, and the parameters relevant for the experiments are presented in Table 3.1. The billing period is 60 minutes.

Table 3.1 VM types used in the experiments

<table>
<thead>
<tr>
<th>Type</th>
<th>Memory (GB)</th>
<th>Core Speed (ECU)</th>
<th>Cores</th>
<th>Cost($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.small</td>
<td>1.7</td>
<td>1</td>
<td>1</td>
<td>0.06</td>
</tr>
<tr>
<td>m1.medium</td>
<td>3.75</td>
<td>2</td>
<td>1</td>
<td>0.12</td>
</tr>
<tr>
<td>m1.large</td>
<td>7.5</td>
<td>2</td>
<td>2</td>
<td>0.24</td>
</tr>
<tr>
<td>m1.xlarge</td>
<td>15</td>
<td>2</td>
<td>4</td>
<td>0.48</td>
</tr>
<tr>
<td>m3.xlarge</td>
<td>15</td>
<td>3.25</td>
<td>4</td>
<td>0.50</td>
</tr>
<tr>
<td>m3.xxlarge</td>
<td>30</td>
<td>3.25</td>
<td>8</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Four workflow applications are used in these tests. They are Montage (generation of sky mosaics), Cyber Shake (earthquake risk characterization), LIGO (detection of gravitational waves), and SIPHT (bioinformatics). These applications are characterized (Juve et al. 2013).

The average values observed for 50 executions are presented along with the standard deviation. In most scenarios, EAIPR drastically reduces the execution time of applications compared to EIPR and IC-PCP. The results show up to 43 percent of execution time reduction compared to the execution time
of the IC-PCP algorithm. Figure 3.3 also shows the effect that increased replication budget on the applications execution time.

![Normalized average execution time of applications for different application sizes](image)

**Figure 3.3 Normalized average execution time of applications for different application sizes**

In most cases utilization of opportunistic replication (i.e., utilization of the available gaps in the allocated VMs, without deployment of extra VMs) introduces the performance improvements in the application execution time. As expected, the increased replication budget tends to further reduce the execution time, although the amount of performance improvement is application-dependent. For example, the CyberShake large workflow experiences significant performance gains with budget increment. The resource utilization percentages observed for 50 executions are presented along with the standard deviation. In most scenarios, EAIPR drastically increases the resource utilization percentage for all the applications compared to EIPR and IC-PCP. The results increase up to 30-35 percent of resource utilization in IC-PCP especially for CyberShake and they are illustrated in Figure 3.4.
Figure 3.4  Resource utilization of applications for different application sizes

The average values observed for 50 executions are presented along with the standard deviation in Table 3.2.

Table 3.2 Average Cost (U$) utilization of the EIPR and EAIPR

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Size</th>
<th>ICPCP</th>
<th>EIPR</th>
<th>EAIPIR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Budget =0</td>
<td>Budget =0.5X</td>
</tr>
<tr>
<td>Montage</td>
<td>Medium</td>
<td>0.06 (0)</td>
<td>1.44 (0.19)</td>
<td>2.14 (0.29)</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>0.06 (0)</td>
<td>3.25 (0.34)</td>
<td>3.25 (0.34)</td>
</tr>
<tr>
<td></td>
<td>Verylarge</td>
<td>20.09 (18.81)</td>
<td>22.53 (19.92)</td>
<td>61.54 (11.13)</td>
</tr>
<tr>
<td>Cybershake</td>
<td>Medium</td>
<td>0.47 (0.46)</td>
<td>0.38 (0.06)</td>
<td>0.56 (0.08)</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>1.13 (0.14)</td>
<td>0.92 (0.16)</td>
<td>1.38 (0.15)</td>
</tr>
<tr>
<td></td>
<td>Verylarge</td>
<td>46.22 (7.2)</td>
<td>58.86 (4.16)</td>
<td>88.26 (6.24)</td>
</tr>
<tr>
<td>LIGO</td>
<td>Medium</td>
<td>0.83 (0.13)</td>
<td>1.31 (0.75)</td>
<td>1.88 (1.09)</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>2.1 (0.24)</td>
<td>2.86 (0.88)</td>
<td>4.23 (0.81)</td>
</tr>
<tr>
<td></td>
<td>Verylarge</td>
<td>41.26 (17.69)</td>
<td>23.05 (20.79)</td>
<td>39.54 (26.19)</td>
</tr>
<tr>
<td>SIPHT</td>
<td>Medium</td>
<td>0.76 (0.06)</td>
<td>1.09 (0.12)</td>
<td>1.44 (0.14)</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>1.22 (0.09)</td>
<td>1.55 (0.19)</td>
<td>2.14 (0.22)</td>
</tr>
<tr>
<td></td>
<td>Verylarge</td>
<td>14.51 (0.96)</td>
<td>16.32 (1.56)</td>
<td>21.75 (1.85)</td>
</tr>
</tbody>
</table>
In most scenarios, EAIPR drastically reduces the execution time of applications compared to EIPR and IC-PCP for larger application sizes. IC-PCP and EIPR algorithm do not enable the replication of tasks across multiple clouds. The proposed system EAIPR considers the behaviour of cloud resources during the scheduling process and also applies replication of tasks to increase the chance of meeting application deadlines. The results show the increase of 40 milliseconds compared to the execution time of the IC-PCP algorithm. Figure 3.5 also shows the effect that the increased replication budget has on the application execution time.

![Graph showing execution time comparison](image)

**Figure 3.5** Normalized average execution time of applications for large application sizes

### 3.4 SUMMARY

The research method has presented a novel workflow scheduling algorithm for utility grids, known as the IC-PCP with ABC algorithm that tries to make a schedule, which reduces the complete execution cost of a workflow, when fulfilling a user-defined deadline. EAIPR algorithm, utilizes idle time of the provisioned resources to imitate the workflow tasks in order
to alleviate the consequences of performance deviation of resources with the intention that the soft deadlines could be attained. With the aim of decreasing the influence of performance deviation of public cloud resources in the targets of workflows, a novel algorithm known as EAIPR is presented. That algorithm considers the conduct of cloud resources for the period of scheduling process and as well as it employs the imitation of tasks for enhancing the possibility of meeting application targets. EAIPR scheduler has the capability of selecting the default storage capacity in virtual machines as well as additional storage. The research method has concentrated on the reduction of cost by aiming to reach the application target and has not reached the identification of execution time of every task in the workflow. Moreover, it contains greater difficulty with security.