Secure Scheduling of Scientific Workflows in Cloud

Shubhro Roy¹, Arun Ramamurthy¹, Anand Pawar²,∗, Mangesh Gharote¹ and Sachin Lodha¹

¹TCS Research, Tata Consultancy Services, India
²Indian Institute of Information Technology, Pune, India

Keywords: Scientific Workflow Scheduling, Cloud Computing, Security in Cloud, Task Scheduling, Evolutionary Optimization Algorithms.

Abstract: Scheduling of tasks in scientific workflow has been challenging due to heterogeneous and interdependent tasks in workflow. The scheduling involves selection of different types of virtual machines (VM) belonging to different instance series (computing, memory, storage) to minimize the overall execution cost and time (makespan). Apart from VM selection, selection of security services (such as authentication, integrity, confidentiality) is critical. In this paper, we propose OptReUse - a workflow schedule generation algorithm for efficient reuse of VMs. Our approach of OptReUse algorithm along with combinatorial optimization approach gives lower cost of scheduling compared to the prior art without incurring delay in the makespan. Further, we enhance the security model by accurate estimation of risks. Our experiments using standard scientific workflows demonstrate that the proposed method gives lower costs compared to the prior VM resource utilization methods.

1 INTRODUCTION

Scientific workflow consists of several heterogeneous and interdependent computational tasks. Such workflows could occur in various research areas, such as in physics, bioinformatics, seismology, industrial control systems, etc. Large-scale computational infrastructure is needed to schedule them. Scheduling involves processing the tasks on selected computational resources considering the dependency order. Since Cloud Service Providers (CSPs) offer highly scalable computational resources, such as Virtual Machines (VMs) at pay-per-use model, cloud has emerged as a cost- and time-effective platform for solving scientific workflow problems.

CSPs offer different series of VMs such as computation intensive, memory intensive and data intensive, for processing heterogeneous tasks. Each series of VMs contains different VM types which vary in renting cost and processing capacity. Higher processing capacity VMs help in lowering the workflow execution time (makespan) but may lead to higher rental cost. While lower processing capacity VMs could be cheaper, processing all tasks on them might lead to violation of deadlines. So, selecting the right VM combination for heterogeneous tasks becomes a challenging combinatorial optimization problem.

2 RELATED WORK

In the study (Liu et al., 2020) present a detailed survey on scientific workflow scheduling algorithms. Finding the right combination of VMs is a challenging combinatorial optimization problem (Zhou et al., 2019; Mboula et al., 2020). As this problem is NP-Hard (Hilman et al., 2018), researchers have proposed different evolutionary optimization algorithms such as Genetic Algorithm (GA) (Shishido et al., 2018), Particle Swarm Optimization (PSO) (Li et al., 2016; Shishido et al., 2018), Improved PSO (Peng and Wolter, 2019), Frog Leaping Algorithm (Kaur and Mehta, 2017), Firefly based approach (Adhikari et al., 2020), and deadline constrained co-evolutionary GA-based algorithm (Liu et al., 2017).

Evolutionary optimization algorithms are used to determine the optimal VM combinations for processing the tasks in the workflow. A separate Workflow Schedule Generation (WSG) algorithm is required to efficiently schedule each of the tasks on the selected VM combination. WSG algorithm would compute the: workflow execution cost and makespan based on...
the selected VM combinations. Further, researchers proposed different VM resource utilization strategies to reduce the workflow execution cost. In the study (Lee et al., 2015) proposed a resource utilization strategy by delaying a combination of few tasks. This strategy resulted in increase in makespan while achieving the cost benefits. In other study (Malawski et al., 2015) used a similar strategy of delaying tasks to improve resource utilization. They proposed a decision algorithm for scheduling workflows, considering budget and deadline constraints. However, if increase in makespan resulted in violation of deadline, then new VMs with higher processing capacity were selected, which resulted in higher rental cost.

Further, (Li et al., 2016) and (Shishido et al., 2018) proposed a WSG algorithm where VMs were reused among different tasks and cost benefits were achieved without delay in makespan. Thus in summary, prior art has addressed different heuristics and evolutionary algorithms (Challita et al., 2017) for better resource utilization but most have not included the concept of VM reuse or do not explore all possible VM reuse options. In this paper, we propose OptReUse algorithm that overcomes the above stated limitations, and obtain higher cost savings compared to WSG. Specifically, we reduce the cost by exploring all possible VM reuse options, and by not including any compulsory delay in task. Other VM utilization strategies proposed by (Weng et al., 2016) are by monitoring overheads. Further, (Ramamurthy et al., 2021) addresses the problem of VM allocation for scientific workflow considering multi-objectives and uncertainties.

Another critical factor in scheduling of scientific workflows in cloud is security. The security in cloud is a shared responsibility (Kumar et al., 2018). CSPs provide security services to mitigate different attacks. We focus on three crucial security attacks alteration, snooping, and spoofing attacks. Alteration affects the data integrity, snooping affects confidentiality and spoofing provides illicit access to sensitive data. To mitigate these attacks, different cryptographic algorithms are used for task security (Li et al., 2016; Shishido et al., 2018). However, cryptographic algorithms that provide stronger security have higher overheads in terms of computation time, which increase the makespan and operational cost. Further, all the tasks in workflow, might not require the same level of security services. Thus, the problem of scheduling of scientific workflow involves selecting the right combinations of VMs and security model (i.e. different security services and level) for keeping the risk rate below permissible limit.

The security model proposed in the prior art (Li et al., 2016) considers that each task would require cryptographic security services for a limited period, more specifically, for at most one hour. But tasks can continue to use a VM for several hours till the processing is complete. Hence, risk is underestimated in the security model in prior art (Li et al., 2016). We propose an enhanced security model that keeps the risk rate of the workflow below a permissible limit while accurately estimating the risk. Thus, the main contributions of the paper are:

- OptReUse algorithm for efficient reuse of VMs to obtain cost benefits, and
- An enhanced security model, and more accurate estimation of risks.

The rest of the paper is organized as follows: Section 3 describes the problem and the system model. Section 4 compares the WSG and OptReUse algorithms. Section 5 presents coding strategies for GA and PSO and contains the experimentation results. Conclusion and future works are laid out in Section 6.

3 WORKFLOW MODEL

A scientific workflow is modeled as a Directed Acyclic Graph (DAG) and represented as $DAG(T,E)$, where $T$ represents the set of tasks $\{t_0, t_1, t_2, ..., t_n\}$ modeled as vertices and $E$ represents the set of edges $\{e_0, e_1, ..., e_k\}$ modeled as dependencies between the tasks. The predecessor and successor of a task $t_i$ is represented as $pre(t_i)$ and $suc(t_i)$. A task $t_i$ cannot start until all its predecessor tasks have completed their execution. A sample workflow is shown in Figure 2.

CSPs provide different series of VMs for processing heterogenous tasks. We consider three series of VMs namely Computing Optimized, Memory Optimized and Storage Optimized (refer Tables 2, 3 and 4 from (Li et al., 2016)). Each VM is rented based on an hourly pricing model. For example, if a task has a processing time of 1 hour and 10 minutes on a VM $v_{m_i}$, the user must pay for 2 hours. We assume there is no bound on the number of VMs that can be rented from a CSP.

The scientific workflows running on cloud are vulnerable to different types of security attacks. In this work, we have considered three types of attacks: 1) Snooping attack (theft of information), 2) Alteration attack (modification of information), 3) Spoofing attack (deceitful access to information). To protect the VMs against these attacks, security services such as authentication, integrity and confidentiality are used.
Different security algorithms under these security services are listed in (Xie and Qin, 2006) (Tables 1, 2 and 3), (Li et al., 2016) (Table 5). As discussed in these works, the algorithms differ in the offered security level and the associated overhead. In general, an algorithm with a higher security level has a higher overhead than the one with a lower security level. Hence, using lower levels of security services can reduce cost and makespan but increases the attack probability and vise-versa. The security overheads for the three services (a, g, c) are computed as given in (Li et al., 2016) and (Xie and Qin, 2006). The overall security overhead for task \( t_i \) is the sum of individual security overheads as below:

\[
TSC(t_i) = SC^a(t_i) + SC^g(t_i) + SC^c(t_i)
\] (1)

The security overhead significantly increases the task processing time and leads to higher task execution cost. Hence, optimal selection of security levels for each security service is essential for minimizing workflow execution cost and makespan.

The pictorial representation of task processing on VMs is obtained from (Li et al., 2016) (Figure 2) and the equations for task execution process analysis are modified in our paper. For processing a task \( (i, t_i) \), the output data (GB) from the VM of its predecessor task \( (say, t_j) \) needs to be transferred to the VM of \( t_i \). The total input data transfer time for task \( t_i \) is given as

\[
TT(t_i) = \sum_{t_j \in \text{pre}(t_i)} d_{ij}^p / B
\] (2)

Note that \( d_{ij}^p = 0 \) if \( t_j \) reuses the same VM instance type allocated to \( t_j \) since data transfer is not required when a VM is reused. The transfer time for start tasks is assumed to be zero.

The execution time and total processing time (including all security overheads) for a task is given by

\[
ExT(t_i, vm_k^i) = \sum_{t_j \in \text{pre}(t_i)} d_{ij}^p / B
\] (3)

\[
PT(t_i, vm_k^i) = TT(t_i) + ExT(t_i, vm_k^i) + TSC(t_i)
\] (4)

The start time and end time for a task is related as

\[
ET(t_i) = ST(t_i) + PT(t_i, vm_k^i)
\] (5)

Total Execution Cost for the entire workflow can be also computed as

\[
TEC = \sum_{i=1}^{n-1} \left( (ET(t_i) - ST(t_i) - IT_a(t_j, vm_k^i))/60 \right) c_k^i
\] (6)

Since VMs are borrowed on hourly basis, the idle time exists between the completion of the task and the end of that hour slot. The Total Execution Time (\( TET \)), that is, the makespan of a workflow, is given by the equation as

\[
TET = \max\{ET(t_i) | t_i \in T\}
\] (7)

### Table 1: Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition and units</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_i )</td>
<td>An individual task of a workflow</td>
</tr>
<tr>
<td>( vm_k^i )</td>
<td>VM of instance series s and type k</td>
</tr>
<tr>
<td>( p_k^i )</td>
<td>Processing Capacity (MFLOPS)</td>
</tr>
<tr>
<td>( c_k^i )</td>
<td>Renting cost ($/hr)</td>
</tr>
<tr>
<td>( q = {a, g, c} )</td>
<td>Set of security services</td>
</tr>
<tr>
<td>( a, g, c )</td>
<td>Authentication, Integrity, Confidentiality</td>
</tr>
<tr>
<td>( sr )</td>
<td>Required security levels</td>
</tr>
<tr>
<td>( sp )</td>
<td>Provided security level</td>
</tr>
<tr>
<td>( SC^a(t_i) )</td>
<td>Individual security overhead</td>
</tr>
<tr>
<td>( TSC(t_i) )</td>
<td>Total security overhead</td>
</tr>
<tr>
<td>( d_{ij}^p )</td>
<td>Output data from predecessor task</td>
</tr>
<tr>
<td>( B )</td>
<td>Bandwidth of data transfer (GB/s)</td>
</tr>
<tr>
<td>( TT(t_i) )</td>
<td>Total input data transfer time (min)</td>
</tr>
<tr>
<td>( W_t )</td>
<td>Task workload (MFLOP)</td>
</tr>
<tr>
<td>( ExT(t_i, vm_k^i) )</td>
<td>Task execution time on ( vm_k^i )</td>
</tr>
<tr>
<td>( PT(t_i, vm_k^i) )</td>
<td>Total task processing time on ( vm_k^i )</td>
</tr>
<tr>
<td>( ST(t_i), ET(t_i) )</td>
<td>Start and End time of task (hr)</td>
</tr>
<tr>
<td>( TEC )</td>
<td>Total Execution Cost ($)</td>
</tr>
<tr>
<td>( IT_a(t_j, vm_k^i) )</td>
<td>Available idle time on ( vm_k^i ) (min)</td>
</tr>
<tr>
<td>( TET )</td>
<td>Total Execution Time (hr)</td>
</tr>
</tbody>
</table>

### 3.1 Risk Analysis

Different security services are provided to mitigate risk of attacks. It could happen that, there is difference between the security services required and services provided for a particular workflow. Hence, the risk of attack exists.

The security model considered in the prior art (Li et al., 2016) assumed the risk rate over a unit time interval could be an hour) for which \( T_i \) is executed on the VM. For a given task \( t_i \), the arrival rate \( \lambda_i \) is the number of time intervals (hours) for which \( t_i \) is executed on the VM. The risk probability of attack for a given task \( P(t_i, sl_i^j) \) is given as

\[
P(t_i, sl_i^j) = 1 - exp(-\lambda_i (sr_i - sp_i) N(t_i))
\] (8)

where, \( N(t_i) \) is the number of time intervals (each time interval could be an hour) for which \( t_i \) is executed on the VM. For a given task \( t_i \), the arrival rate \( \lambda_i \) of snooping, alteration and spoofing attacks is assumed to follow a Poisson distribution. Consequently, the risk probability \( P(t_i) \) for task \( t_i \), involving all three security services is computed as

\[
P(t_i) = 1 - \prod_{l \in \{a, g, c\}} (1 - P(t_i, sl_i^l))
\] (9)

For a given a workflow, consisting of a set of \( T \) tasks, risk probability \( P(T) \) can be computed as

\[
P(T) = 1 - \prod_{t_i \in T} (1 - P(t_i))
\] (10)

This value of \( P(T) \) must be lower than the risk rate threshold \( P_r \ (P_r \in [0, 1]) \), which is the permissible risk
rate of the workflow. The constraint \( P(T) \leq P_c \), can be also written as \( 1 - P(T) \geq 1 - P_c \).

\[
\prod_{t \in T} (1 - P(t_i)) \geq 1 - P_c
\]

On further expanding the LHS

\[
\prod_{t \in T \in \{a,g,c\}} (1 - P(t_i, sl'_i)) \geq 1 - P_c
\]

of

\[
\prod_{t \in T \in \{a,g,c\}} \exp(-\lambda(sl'_i - sl_l)(N(t_i))) \geq 1 - P_c
\]

Taking \( \log \) on both sides the inequality becomes:

\[
\sum_{t \in T \in \{a,g,c\}} \sum_{l} -\lambda(sl'_i - sl_l)N(t_i) \geq \log(1 - P_c)
\]

By introducing the correction factor \( N(t_i) \), the security model is enhanced and it provides a better estimation of risk. However, providing security levels to tasks lower than required could result in violation of risk rate constraint whereas higher security levels could result in high cost and makespan. This makes the workflow scheduling problem challenging.

### 4 OUR APPROACH

The flow chart in Figure 1 shows our methodology for scientific workflow scheduling. As stated earlier, we use evolutionary optimization algorithm (GA or PSO) to obtain a best combination of VM types and security levels for tasks in a workflow. Further, an efficient workflow schedule generation is required to allocate the selected VMs for lowering the cost and the makespan. The efficient workflow schedule is generated using OptReUse algorithm, which does the VM reuse along with the schedule generation. The process stops, when converge criteria is met (i.e. best solution is obtained).

**Problem Statement:** Given a scientific workflow with \( n \) tasks, the problem is to determine the optimal combination of VM types and security levels for each task such that the overall execution cost is minimal, and the workflow is processed within a given deadline \( D \) and permissible risk limit.

**Objective:**

\[
\text{Minimize } TEC
\]

Subject to:

\[
TET \leq D
\]

\[
P(T) \leq P_c
\]

Since scientific workflow scheduling problem is a combinatorial optimization problem and NP-Hard (Li et al., 2016), we use evolutionary optimization algorithm for solving the problem. We demonstrate the results using both the GA and PSO. In the below section, first we illustrate the benefits of VM re-utilization and in the subsequent section, discuss the intuition behind our workflow schedule generation OptReUse algorithm.

#### 4.1 Intuition: WSG and OptReUse

If each task is allocated to a separate VM then it leads to under utilization of the resources. For the task with similar instance type, VM can be reused, resulting in reduced rental costs and lower data transfer delays. WSG proposed by (Li et al., 2016), (Shishido et al., 2018) helps to reduce the cost and transfer time delays by reuse of VMs among adjacent tasks. For example, consider the workflow instance as shown in Figure 2. The tasks are traversed by a traversal order, such as topological sort \((t_0, t_1, t_2, t_3)\), and simultaneously allocated to VMs based on the VM types selected by the evolutionary optimization algorithm. This enables \( t_3 \) to reuse the VM allocated to \( t_0 \) and avoid extra data transfer.
transfer delays.

For \( t_0 \) and \( t_2 \), the total cost to be paid is for 150 minutes (i.e. 3 hours) and for the remaining tasks new VMs need to be borrowed. Task \( t_2 \) which belongs to the same VM instance series (Data intensive), a new VM is borrowed for 125 mins (i.e. 3 hours). The WSG approach proposed in prior art (Li et al., 2016), (Shishido et al., 2018) keeps the VM reuse limited between adjacent tasks only.

The VM on which \( t_2 \) is processed has an idle time of 30 minutes \((180 - (70 + 80))\). Task \( t_2 \) which is non-adjacent task to \( t_3 \) can also avail this idle time. Note, task \((t_0, t_3 \) and \( t_2 \)) belongs to the same instance series. But as task \( t_2 \) starts only after completion of task \( t_1 \) (i.e \( ST = 160 \) mins). The available idle time on VM for \( t_2 \) is 20 minutes, which can be utilized by task \( t_2 \). Hence, the VM rental cost for task \( t_2 \) has to be paid only for 105 minutes \((125 - 20)\) (i.e. 2 hours), which is lower than the cost paid for \( t_2 \) using WSG approach. Our proposed OptReUse algorithm is based on this VM reuse strategy. Along with this strategy, task ordering has significant impact of VM reuse. In the subsequent section, we elaborate on how ordering of task belonging to the same VM instance series can further reduce the VM rental cost.

### 4.2 Task Order Selection and VM Reuse

Suppose, tasks \( t_0, t_1 \) and \( t_2 \) are part of a workflow, as displayed in Figure 3. Assume, these tasks belong to the same VM instance series (data intensive). Consider that the least expensive VM types are selected for these tasks. The first workflow in Figure 3 displays, task \( t_1 \) reuses the VM and in second workflow task \( t_2 \) reuses the VM of task \( t_0 \).

- If \( t_1 \) reuses the VM of \( t_0 \), then the total cost to be paid for \( t_0 \) and \( t_1 \) is for 150 minutes (i.e. 3 units) and the cost for \( t_2 \) has to be paid for 130 minutes (i.e. 3 units). Thus, the total cost to be paid for all the three tasks is 6 units.

- If \( t_2 \) has reused the VM of \( t_0 \), then the total cost to be paid for \( t_0 \) and \( t_2 \) is for 170 minutes (i.e. 3 units) and the cost for \( t_1 \) is for 110 minutes (i.e. 2 units). Thus, the total cost for second workflow is 5 units.

Thus, task order impacts the VM reuse and cost. In our approach, along with VM re-utilization across adjacent and non-adjacent task, benefits due to task ordering is considered.

#### 4.3 OptReUse Algorithm

The best combination of VM types and security levels for workflow tasks are determined by an evolutionary optimization algorithm (like GA or PSO). Different combinations are explored to obtain the best combination using evolutionary optimization algorithm. Each combination is represented as a chromosome (GA) or particle (PSO). The OptReUse algorithm computes the execution cost (\( TEC \)), time (\( TET \)) and risk rate (\( P(T) \)) for each such combination to determine the fitness value. OptReUse algorithm comprises of two parts: Algorithms 1 and 2.

Using Algorithm 1, we compute the start time and the end time of each task, for the given combination of VM types and security levels. The other inputs to Algorithm 1 are processing capacity, workload, output data size of tasks, which assist in computing the \( ST \) and \( ET \). The \( ST \) of a task is same as the \( ET \) of its predecessor task which completed last. First, we do a general sort on the \( ST \) array in an ascending order. Later, using the task order selection vector provided by the evolutionary algorithm, we rearrange the task. These task vector comprises of those task having the same start time.

**Algorithm 1: Start Time Computation and Task Ordering.**

1: INPUT: VM and security services for \( n \) tasks
2: INPUT: Task order vector based on start time
3: INPUT: Processing capacity, workload, output data size, VM rent cost
4: for \( i = 1, 2, \ldots, n \) tasks do
5: \( PT[i] \) for each task.
6: end for
7: for \( i = 1, 2, \ldots, n \) tasks do
8: \( ST[i] = \max\{ET[j] ; j \in pre(i)\} \)
9: Calculate \( ET[i] \)
10: end for
11: Sort \( ST \) based on the task order selection \( ST \)
12: OUTPUT: \( ST, ET \)

The output of Algorithm 1 is used as input to Algorithm 2. Before allocating a task \( i \) to a new VM, all possible underutilized VMs of the same instance series and type are explored for reuse.
- If the task reuses a VM used by an adjacent task then reduction in cost is due to (i) re-utilization of available idle time on the VM and (ii) data transfer costs between the tasks is null.
- If the task reuses a VM used by a non-adjacent task then reduction in cost is only due to re-utilization of available idle time on the VM.

If such a reusable VM is found, the task \( i \) is allocated to the VM where it gets maximum cost reduction or else a new VM is rented for that task.

Algorithm 2: Task-VM Allocation Algorithm.

1: INPUT: \( ST, ET \)
2: Initialize \( TEC = 0 \)
3: Initialize \( IT[n] = \{0, 0, \ldots, 0\} \)
4: for \( i = 1, 2, \ldots, n \) tasks do
5:   Search for \( vm^i_k \) where idea time is available.
6:   if VM is available then
7:     Allocate task \( i \) to VM, where maximum cost reduction is available.
8:     Update Idle time on \( vm^i_k \), if reused.
9: else
10:   Allocate a new VM.
11: end if
12: Compute new idle time and update \( IT[i] \).
13: Increment \( TEC \) as per equation 6.
14: Update \( ST \) and \( ET \).
15: Sort \( ST \).
16: end for
17: Calculate \( TET = \max(ET) \) as per equation 7.
18: Calculate \( P(T) \) as per equation 10
19: OUTPUT: \( TEC, TET, P(T) \).

Since OptReUse does not compute \( TEC \) while traversing the workflow DAG by a traversal order, but sorts tasks based on their start time and allocate resources, OptReUse can find VM reuse between tasks of different workflows running simultaneously. The implementation details and experimentation results on various scenario is explained in the subsequent section.

5 IMPLEMENTATION DETAILS

The individuals in the population-based evolutionary algorithms represent a combination of VM types, security levels and task order. Workflow schedule generation approaches like WSG or OptReUse helps in efficient resource utilization for a given combination. Each combination is used as an input to the WSG and OptReUse algorithms. The total cost, makespan and risk rate for that combination is computed. The fitness of a chromosome or a particle is measured in terms of total cost, makespan and risk rate. In the subsequent section, we discuss the individual coding strategy and the implementation details for both the evolutionary algorithms.

Parameters and Coding Strategy: In GA we used random sampling with Tournament Selection. Simulated binary crossover (\( prob = 0.9 \)) operation and polynomial mutation operation were adapted. The chromosome coding strategy is shown in Figure 4. The first chromosome consists of set of \( n \) values for VM type, authentication, integrity and confidentiality service level for each task. The second chromosome is for task order selection. These tasks are those having the same starting time. The second chromosome represents the task ordering between tasks having same starting time. The PSO particle coding strategy is similar to GA chromosome coding strategy. Both GA and PSO is run for 1000 generations and the number of particles in each generation is 150. For PSO, the initial velocity is kept 0. Initially parameters have values like \( w = 0.64 \), \( c_1 = 2.0 \) and \( c_2 = 2.0 \). However, the parameters are adaptive which means they keep changing at each iteration.

5.1 Results and Discussion

Our approach is tested on three standard scientific workflows: LIGO, SIPHT, CyberShake (refer Figure 7 from (Li et al., 2016)). The experimentation settings are considered from the paper (Li et al., 2016). The workload and output data for each task is randomly generated from a uniform distribution in the range [5000, 50000] GFLOP and [10, 100] GB, respectively. The bandwidth is considered as 0.1 GB/s. The two evolutionary optimization algorithms GA and PSO are used for selecting the optimal combination of VM types, security levels and task order for tasks. The cost performance of our proposed OptReUse algorithm is compared with the workflow scheduling generation (WSG) proposed in (Li et al., 2016).
5.2 Comparison: WSG and OptReUse

We compare the performance between WSG and OptReUse for different risk rates. The results are shown in Figures 5 and 6. We consider an instance of each standard workflow. For a given evolutionary algorithm (GA or PSO) and for a given value of ($P_c$), OptReUse always results in lower cost compared to WSG.

1. For LIGO, the average percentage reduction in cost obtained by OptReUse over WSG is 8.07%.
2. CyberShake is a data intensive workflow requiring mostly storage optimized VMs and they are the costliest VMs. Therefore, in CyberShake workflow more VMs are reused compared to LIGO. Hence, the average percentage reduction in cost is 11.2%.
3. SIPHT workflow is a computation intensive workflow requiring mostly the cheapest compute optimized VMs. But it has the highest number of tasks which makes it possible for OptReUse to search for more reuse cases. We observed for SIPHT the average percentage reduction in cost as 13.15%.

With increase in value of $P_c$, TEC decreases because of the selection of lower security levels. The makespan in all the three benchmark test cases for all the $P_c$ values in the range $[0, 1, ..., 0.9]$ does not change significantly {LIGO = 45.47, SIPHT = 53.81, CyberShake = 102.2} minutes. Note, as time is not an objective function in our problem statement but it is a constraint by a deadline.

5.3 OptReUse: Average Performance

To demonstrate the average performance of our approach, we experimented over number of instances. We generated 75 workflow instances of CyberShake work. Each instance having different task workload and output data size. We chose three (high, medium, low) permissible risk rates ($P_c$) to observe the impact of security levels on workflow execution cost and makespan. The results are demonstrated in Figure 7. Each bar in the figure gives the mean cost and margin of error for confidence interval of 95%. It is observed that OptReUse gives lower mean cost and margin of error compared to WSG. When $P_c$ is increased, the difference between the required and the provided security levels, increases. Hence, with the selection of lower security levels (with lower overheads), overall cost is reduced (refer Figure 7). Note, the average makespan for both WSG and OptReUse algorithm is 46.66 minutes. Therefore, we can conclude that using OptReUse approach, there is reduction in cost without any delay.

5.4 OptReUse: Multiple Workflows

Consider the case of scheduling more than one workflow instance simultaneously. We demonstrate using two instances of CyberShake workflow. Suppose, all the tasks are assigned with the least expensive VM types. Scheduling multiple workflows simultaneously by OptReUse results in lower cost compared to the combined cost of scheduling workflows separately (both by WSG or OptReUse). This is due to reuse of VMs between tasks of different workflow instances as shown in Table 2.
Table 2: Workflow scheduling results.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Approach</th>
<th>Cost ($)</th>
<th>Makespan (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WSG</td>
<td>73.56</td>
<td>46.72</td>
</tr>
<tr>
<td>2</td>
<td>WSG</td>
<td>67.66</td>
<td>51.125</td>
</tr>
<tr>
<td>1</td>
<td>OptReUse</td>
<td>72.876</td>
<td>46.72</td>
</tr>
<tr>
<td>2</td>
<td>OptReUse</td>
<td>67.35</td>
<td>51.125</td>
</tr>
<tr>
<td>1&amp;2</td>
<td>OptReUse</td>
<td>137.45</td>
<td>51.125</td>
</tr>
</tbody>
</table>

6 CONCLUSION

Scientific workflow scheduling is about selecting the right VMs, security levels and schedule generation such that the overall cost and makespan is minimal. We demonstrate, how VM reuse can reduce the overall cost without any delay. In particular, we demonstrate the benefits of VM reuse across adjacent and non-adjacent task, and further due to ordering of tasks with the same start time. We design an enhanced security model for accurate estimation of risk. The results are shown by using two evolutionary algorithms (GA and PSO) and the approach is tested on three benchmark datasets. Our approach provides significant cost reduction via VM reutilization.

REFERENCES


