PeerJ Computer Science

# A fault-intrusion-tolerant system and deadline-aware algorithm for scheduling scientific workflow in the cloud

Mazen Farid[1,2], Rohaya Latip[1,3], Masnida Hussin[1] and Nor Asilah Wati Abdul Hamid[1]

[1] Department of Communication Technology and Networks, Universiti Putra Malaysia, Selangor, Serdang, Malaysia
[2] Faculty of Education-Saber, University of Aden, Saber, Aden, Yemen
[3] Institute for Mathematical Research (INSPEM), Universiti Putra Malaysia (UPM), Selangor, Serdang, Malaysia

## ABSTRACT

**Background:** Recent technological developments have enabled the execution of more scientific solutions on cloud platforms. Cloud-based scientific workflows are subject to various risks, such as security breaches and unauthorized access to resources. By attacking side channels or virtual machines, attackers may destroy servers, causing interruption and delay or incorrect output. Although cloud-based scientific workflows are often used for vital computational-intensive tasks, their failure can come at a great cost.

**Methodology:** To increase workflow reliability, we propose the Fault and Intrusion-tolerant Workflow Scheduling algorithm (FITSW). The proposed workflow system uses task executors consisting of many virtual machines to carry out workflow tasks. FITSW duplicates each sub-task three times, uses an intermediate data decision-making mechanism, and then employs a deadline partitioning method to determine sub-deadlines for each sub-task. This way, dynamism is achieved in task scheduling using the resource flow. The proposed technique generates or recycles task executors, keeps the workflow clean, and improves efficiency. Experiments were conducted on WorkflowSim to evaluate the effectiveness of FITSW using metrics such as task completion rate, success rate and completion time.

**Results:** The results show that FITSW not only raises the success rate by about 12%, it also improves the task completion rate by 6.2% and minimizes the completion time by about 15.6% in comparison with intrusion tolerant scientific workflow ITSW system.

**Subjects** Algorithms and Analysis of Algorithms, Computer Networks and Communications, Distributed and Parallel Computing, Network Science and Online Social Networks, Security and Privacy

**Keywords** Cloud computing, Fault tolerance, Intrusion tolerance, Reliability, Scheduling scientific workflow

Submitted 26 July 2021
Accepted 24 September 2021
Published 2 November 2021

Corresponding author
Mazen Farid,
mazenfareed7@yahoo.com

Academic editor
Kathiravan Srinivasan

Additional Information and Declarations can be found on page 17

DOI **10.7717/peerj-cs.747**



## INTRODUCTION

Cloud computing technology has become one of the most popular systems for providing end-users with computing services. An important feature of this paradigm is that the resources given can be accessed as a utility where consumers can pay for the services they

use (*Aslam et al., 2017*; *Chang & Wills, 2016*; *Farid et al., 2020*; *Ferdaus et al., 2017*; *Sun et al., 2018*). In order to operate cloud-based applications in an economically-efficient and scalable way, it is highly favorable to deploy large virtual machines (VMs) (*Ala'Anzy & Othman, 2019*; *Ghazouani & Slimani, 2017*; *Rao et al., 2013*).

Scientific computing involves several interdependent intermediate data and subtasks which can be built up by different organizations (*Yuan et al., 2010*). To encourage the automation of complex scientific computational processes, scientific workflows were created (*Rodriguez & Buyya, 2017*). Scientific workflows are handled, controlled and executed by scientific workflow systems (*Lin et al., 2009*) derived from grid computing. With the advancement in cloud computing systems, several researchers are now developing cloud-based scientific workflow systems (*Zhao et al., 2014*).

Although cloud-based scientific workflow systems have several merits, cloud platforms are prone to malfunctions due to their increased functionality and complexity (*Pezoa, Dhakal & Hayat, 2010*; *Yao et al., 2016*). Such defects may have a detrimental impact on the performance of submitted tasks. This is because a system's performance is not only evaluated by the correctness of the measurement results, but also by the time of its availability (*Qin & Jiang, 2006*).

Failure often occurs in the components of a system as the operation fails when a machine bears many loads (*Gupta & Gupta, 2020*). A task failure is a situation where the machine cannot complete the task within a deadline or when the machine ceases to process tasks due to network, memory, or system bugs. As a result of a delay in the completion of one of the tasks caused by a fault, several tasks on other resources may be delayed. To cater to this, many strategies have been proposed. In this paper, we develop a scientific workflow model for fault-intrusion tolerance. A deadline partitioning method determines the completion time for each sub-task.

Cloud platforms follow a multi-tenant coexistence service paradigm. As such, different tenants share the same physical facility using virtualization technologies. The implementation of this model creates flexible control of resources; however, it is associated with risks. There are several vulnerabilities in the virtualized world; an example is VM escape vulnerability (*Wu et al., 2017*). An intruder splits the logical boundaries into side channels and targets members in the same organization (*Zhang et al., 2014*). After the intruder has controlled the entire virtual environment with certain vulnerabilities, he controls the VMs of all tenants (*Szefer et al., 2011*).

Other threats in the cloud include co-residential attacks (*Atya et al., 2017*), side-channel attacks (*Wang et al., 2016*; *Zhang et al., 2014*), and VM escape attacks (*Wu et al., 2017*). A large number of subtasks and intermediate data contained in scientific workflows can easily be targeted by attackers. In addition to researchers' efforts to address threats in the cloud, we propose a fault-intrusion tolerant system and deadline-aware resource provisioning algorithm to protect workflows in clouds. The dynamic task scheduling strategy, based on resource circulation, eliminates latent threats. By regularly deploying and reclaiming VMs, the proposed approach cleans up task executors. The performance of FITSW was tested using WorkflowSim.

Realizing intrusion tolerance is much harder than fault tolerance because it should consider both accidental and malicious faults. Our objective is to achieve the intrusion tolerance of scientific workflows scheduling process.

Our contributions in this study are summarized as follows:

1. We propose a fault and intrusion-tolerant mechanism for scientific workflows (FITSW) by considering the effects of accidental and malicious faults on cloud-based scientific workflows.
2. We develop task executors with various heterogeneous VMs having multiple operating systems.
3. We present a decision-making mechanism that tracks and evaluates the confidence of the intermediate data between sub-tasks during execution.
4. To eliminate latent risks, we suggest a dynamic task scheduling strategy based on recycling resource. FITSW keeps task executors clean by installing and reclaiming virtual machines on a regular basis.

The remainder of this article is organized as follows: "Literature Review" reviews related works. FITSW's principle and threat model are introduced in "Principle of FITSW and Threat Model". "Proposed Scheme" outlines the new scheme. In "Experiments and Results", the experiments and results are discussed, followed by the "Conclusion" that concludes this article.

## LITERATURE REVIEW

Many fault-tolerant algorithms have been proposed in recent decades to reduce the adverse effects of faults in distributed systems. *Javadi et al. (2011)* investigated how failures due to faults can be handled in complex infrastructures. However, the model often involves tracing data failures related to a particular objective, which can be very challenging. *Jhawar & Piuri (2012)* suggested a new dimension in which required fault tolerance properties can be obtained by a third party from applications deployed in cloud systems. For ordinary users, however, it is difficult to choose the appropriate third party. *Zheng et al. (2012)* and *Qiu et al. (2014)* suggested a ranking method in which all components of the cloud were categorized according to invocation structures and invocation frequencies. Using an optimal algorithm based on the ranking results, the fault-tolerant strategies for the various components were computed. However, the exact ranking is difficult as it requires a thorough understanding of the behavior of the target infrastructure as well as long-term trace data of the specific system.

Related to our work which focuses primarily on the study of fault and intrusion tolerance in scientific workflow scheduling, *Yao et al. (2016)* proposed a workflow scheduling algorithm inspired by the immune system. This algorithm can prevent cloud-based scientific workflow disruptions (due to the failure of resources) to protect scientific workflow sub-tasks. To ensure that cloud services are continuously available to defend against security threats, a cloud resource management self-protection solution

was introduced by *Gill & Buyya (2018)*. Centered on the master-slave theorem, *Ding, Yao & Hao (2017)* developed a fault-tolerant scheduling algorithm. For each sub-task, two replicas (one master and one slave replica) are generated and allocated to individual VMs using this algorithm. The ICFWS algorithm, proposed by *Yao, Ding & Hao (2017)* divides the general workflow deadline into sub-task sub-deadlines. Then, based on the assigned sub-deadlines, each scientific workflow task selects an acceptable fault-tolerant strategy using task redundancy and rescheduling strategies.

Cloud services are priced dynamically and are referred to as spot instances of the VMs. Spot cases are cheaper than the VMs offered by the static price scheme. Therefore, *Poola, Ramamohanarao & Buyya (2016)* suggested the use of spot instances to execute scientific workflows in order to minimize cost. *Li et al. (2016)* quantified security for cloud services and thoroughly analyzed the risk rates of scientific workflows. The researchers then created the Security and Cost Aware Scheduling (SCAS) scheme to reduce costs when risk rates are small and deadlines are tight. The cloud-based scientific protective problem in the workflow was formulated as a two-person zero-sum problem by *Wang et al. (2020)*, who suggested the CLOSURE algorithm to confuse adversaries. Nevertheless, if an attacker manages to access a VM and tampers with the scientific workflow to generate an incorrect output, this problem cannot be solved effectively by either of the above works.

Some studies analyze failures to ensure that the workflow is performed successfully even if resources fail. Secret data and sensitive computation also require scientific workflows (*Wang et al., 2019b*). This motivates the need for secure execution of scientific workflows. In this context (*Chen et al., 2017*) used the scientific workflow's slack times to encrypt intermediate data. The encryption algorithm was combined with the task scheduling algorithm to ensure the scientific workflow's confidentiality and reduce cost and time. *Liu et al. (2014)* suggest a security aware intermediate data placement strategy to ensure that intermediate data is secured in three ways: integrity, confidentiality, and privileged access.

In order to increase the availability, confidentiality and integrity of the data of scientific workflow (*Wang et al., 2019b*) used different hash functions, encryption algorithms, and erasure codes while considering scientific workflow deadlines. *Teylo et al. (2017)* studied the scheduling of scientific workflow and the intermediate data allocation. They modeled them as part of an integrated planning challenge to reduce intermediate data transmission in a cloud network. A hybrid evolutionary algorithm known as HEA-TaSDAP was created to optimize task planning, intermediate data allocation strategies and task scheduling. TryXy was created by *Nepal et al. (2017)* and it offers stable scientific workflow storage facilities.

In *Wang et al. (2019a)*, the ITSW designer used a mission replication and voting system to prevent attackers from modifying the results of scientific workflows. But ITSW ignores the delay that can happen because of accidental (*Yao, Ding & Hao, 2017*) or malicious attacks (*Bhattarai et al., 2014*, *2015*) which could lead to an increase in the makespan of the entire workflow. In order to determine sub-deadlines for each sub mission (*Wang et al., 2021*) proposed the INHIBITOR to determine sub-deadlines for each submission by

using the deadline partitioning process. The system for the provision of elastic resources is structured to maximize efficiency and reduce costs, based on these sub-deadlines. This paper deploys the Task-VM mapping framework and elastic resource provisioning mechanism to enhance ITSW intrusion tolerance. Our aim is to develop a FITSW workflow scheduling algorithm that will improve the system's fault and intrusion tolerance, taking into account the delay due to accidental failures or intrusion attacks.

## Scientific workflow security issues in the cloud

It is quite difficult to address the security issues of scientific workflows in clouds due to their unique features. Many scientific workflows are computational intensive (*Li et al., 2016*), thus, they need VMs. These VMs can be targeted easily by attackers (*Narayana & Pasupuleti, 2018*). Executing scientific workflows is also time-consuming (*Yuan et al., 2010*). This provides enough time for attackers to make an intrusion. A scientific workflow is typically a form of Directed Acyclic Graph (DAG) that is extremely vulnerable to attacks, as intermediate errors are inherited in the final result (*Wang et al., 2018*). In addition, the intermediate scientific workflow data also contain sensitive data in some scientific fields. If this data is hacked, users will suffer severe damage (*Wang et al., 2019b*). Adversaries can compromise workflow execution in several ways.

1. The attackers can access and force the VMs that run workflows to go offline.
2. The intruder can gain access to intermediate data without tempering or altering it, rather he delays the finish time of executing sub-tasks (*Yu et al., 2017*). Attacks such as jamming attacks, sniffer attacks, worm propagation, and resource-depletion denial-of-service (*Bhattarai et al., 2014*, *2015*) could be launched to disable the links by congesting the network or monitoring network data flow (*Bhattarai et al., 2015*; *Yu et al., 2017*).
3. In some cases, the purpose of the adversaries is to alter the workflow result rather than interrupt the workflow. This can be achieved through intermediate data workflow manipulation and execution software.
4. The adversaries can also steal the workflow data after breaching the VMs or build a back door for the next attack.

The fault tolerance system generally protects against the first and second types of attacks. However, it cannot prevent the remaining two types. To develop a Fault-Intrusion-tolerant system capable of effectively defending against these four types of threats. There are four major challenges to overcome.

1. To check that each sub-task can be performed without any VM failures, the systems must be able to check the average earliest finish time of the virtual cluster using the sub-task sub-deadline.
2. Systems must be able to (i) assess if workflow's sub-task results are right by checking the confidence of the intermediate data of all replicas and (ii) correct altered outputs to protect the system against the third form of attack by re-executing the current task.

3. The system should be strong enough to withstand the fourth form of attack by removing latent threats and cleaning up executors using resource recycling technique.
4. Instead of security, the efficiency of the system is also very important. Therefore, the system's fault-intrusion-tolerant mechanism should not negatively impact workflow efficiency according to the proposed decision mechanism.

# PRINCIPLE OF FITSW AND THREAT MODEL

## Principle of FITSW

The DAG is an example of a scientific workflow. It is expressed by $G = (T, D)$, where $T = \{t_1, t_2, \ldots, t_n\}$ represents a collection of subtasks for workflow and $D$ is representing the intermediate data between sub-tasks. $d_{i,j} \in D$ refers to intermediate data generated from subtask $t_i$ and used by subtask $t_j$. every workflow sub-task $t_i$ is measured in Million instructions (*MIi*) (*Jhawar & Piuri, 2012*). pred ($t_i$) and succ ($t_i$) represent the predecessors and successors of subtask $t_i$, respectively. Sub-task $t_i$ cannot begin in scientific workflows until all its predecessors are completed.

The workflow executor is one VM in several cloud-based scientific workflow systems. It ensures that every subtask workflow is completed with just one VM. This is highly risky as the adversaries will only jeopardize a single VM to destroy workflow execution. Therefore, a virtual cluster (*Liu et al., 2018*) is suggested for task execution. There are several VMs in the virtual cluster and these VMs run the same workflow subtask in parallel.

Three replicas of each workflow sub-task $t$ are copied into FITSW: $t^{first}$ (the first replica), $t^{second}$ (the second replica) and $t^{third}$ (the third replica), which are executed by three heterogeneous VMs. The heterogeneity of VMs is represented primarily by variations in operating systems. Linux VM, Windows VM and Solaris VM operate for each sub-task workflow. The decision mechanism in FITSW relies mainly on identical results and the sub-deadline of the sub-task. If the confidence of any sub-task is less than 1, the sub-task is re-executed.

## Principle of ITSW-RV

In ITSW-RV, we deploy a random virtual task execution cluster with different VM numbers (between 3–10). The number of replicas for each subtask must be the same for a random number of VMs with different operating systems. The number of similar results is increased in this situation and the ability to receive most of the results is also increased. Our experiment indicates that an increase in the number of VMs contributes to a reduction in the overall algorithm completion time.

## Task-VM mapping

Aside from the resource provisioning strategy, task-VM mapping is an essential step in workflow scheduling also (*Rodriguez & Buyya, 2017*). Various task-VM mapping relationships can be used to achieve various scheduling objectives, such as decreasing the cost of cloud scientific workflows (*Zhou, He & Liu, 2016*), maximizing the use of virtual machines (VMs) during the execution of scientific workflows(*Lee et al., 2015*), and

reducing the makespan for a scientific workflow to complete (*Jiang, Haihong & Song, 2017*). In an actual production environment, the authors suggest in *Sangaiah et al. (2019)* an algorithm for scheduling tasks to be mapped into machines. Similarly, a subtask cannot begin execution before a VM has been allocated. As a result, before using the proposed task-VM mapping algorithm, we will first allocate VMs. Additionally, the system cannot use mapped VMs for the second and third replicas. Finally, for any sub-tasking replica, the HEFT (*Topcuoglu, Hariri & Wu, 2002*) algorithm selects the required VM. The steps taken are described below.

1. To perform tasks, the upward ranking approach is employed as shown in Eq. (1).

$$rank_u(t_i) = \overline{\omega_i} + max_{t_j \in succ(t_i)} \left\{ \frac{d_{i,j}}{BW} + rank_u(t_j) \right\} \tag{1}$$

   The average time available for the VMs to perform $t_i$ is $\omega_i$. By crossing the task graph upward, the rank is recurrently calculated, so it is called upward rank. The upward rank value for the last subtask $t_{last}$ is equivalent to Eq. (2).

$$rank_u(t_{last}) = \overline{\omega_{last}} \tag{2}$$

2. Every sub-task is sorted by a non-increasing $rank_u$ value in a scheduling list.
3. A subtask is chosen in the schedule list and copied into three replicas.
4. For all replicas, $t_i^{first}$, $t_i^{second}$ and $t_i^{third}$, each Earliest Finish Time (EFT) value on VM $vm_j$ is calculated using Eqs. (3) and (4).

Only direct predecessors of a subtask are needed to decide the earliest start time of the three replicas.

$$EST(t_i^x, vm_j) = max\left\{ ready_j, max_{t_m \in pred(t_i)} \left\{ max_{y \in \{first,second,third\}} \left\{ EFT(t_m^y) + \frac{d_{m,i}}{BW} \right\} \right\} \right\}, \tag{3}$$

$x \in \{first, second, third\}$

$$EFT(t_i^x, vm_j) = \frac{MI_i}{vmp_j} + EST(t_i^x, vm_j), x \in \{first, second, third\} \tag{4}$$

$ready_j$ shows the earliest time at which $vm_j$ is ready to execute sub-tasks, $EFT(t_m^x)$ denotes the earliest finish time of $t_m^x$, The size of each sub-task $t_i$ is measured in $MI_i$ (*Jhawar & Piuri, 2012*). $vmp_j$ represents the processing power of $vm_j$ and it is measured in Million instructions per second *MIPS*.

As predicted, a delay with no malicious connections is less than a delay with malicious connections (*Bhattarai et al., 2014*), the finish time of the sub-task must be less than the sub-deadline. Otherwise, intruders would have completed their objectives before the response party performs critical path analysis. In our model, we use a task executor that contains three *VMs* with different operating systems to execute individual sub-task. Hence, to determine the *EFT* of each sub-task and compare it with the sub-deadline, we calculate the average *EFT* of all replicas for each subtask by the following equation.

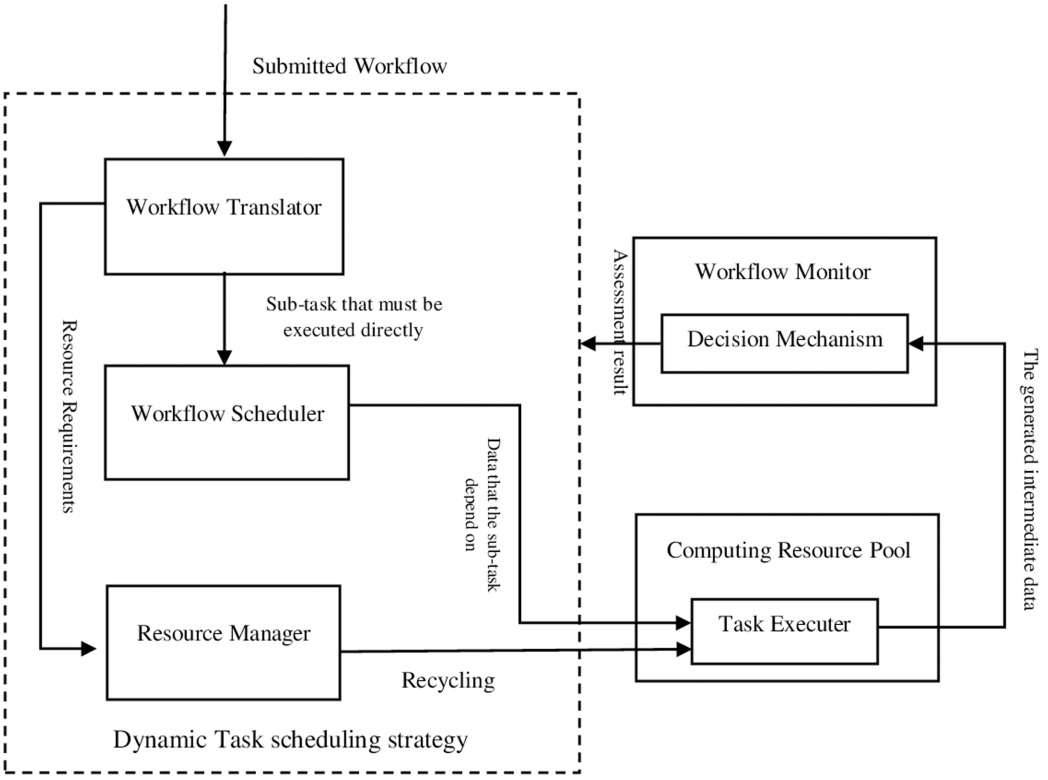


**Figure 1** **The proposed FITSW system.** Full-size DOI: 10.7717/peerj-cs.747/fig-1

$$AEFT(t_i) = \frac{\sum_{x=first}^{third} EFT\left(t_i^x, vm_j\right)}{no.\ of\ vms\ in\ virtual\ cluster} \tag{5}$$

## Proposed Scheme

The outline of the proposed FITSW system is illustrated in Fig. 1. We recommend that multiple heterogeneous *VMs* should be used as task executors to improve the workflow execution fault tolerance. The proposed decision mechanism verifies and evaluates the confidentiality of intermediate data by the executors. It also checks the earliest finish time of each subtask. According to the evaluation of the proposed decision mechanism, a dynamic task scheduling strategy is applied based on resource circulation. It decides whether to implement or recycle task executors. This approach removes inherent threats and cleans the environment for executing scientific workflows.

## Proposed decision mechanism

Attackers mainly aim at controlling the outcome of a workflow rather than terminating it. This could lead to a false workflow output. If the task executer is compromised by this type of attack, the sub-tasks will have multiple results. A decision mechanism to turn multiple inputs into one output can be applied in order to avoid this threat, thereby

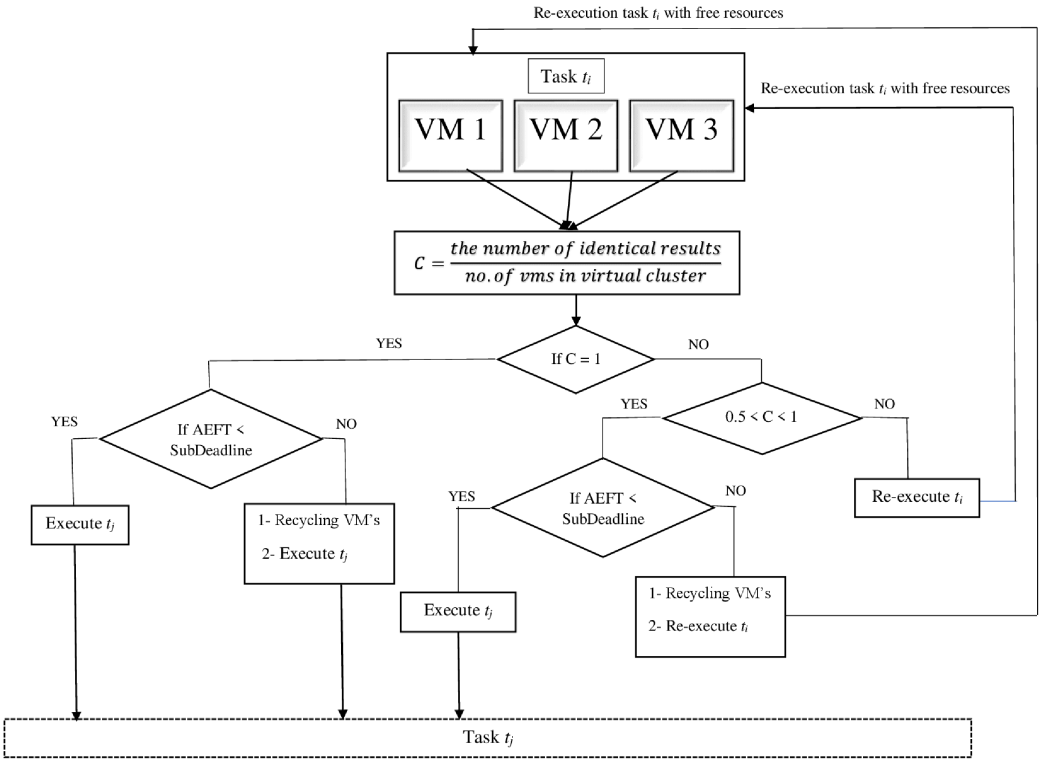


**Figure 2** **Decision mechanism.** Full-size DOI: 10.7717/peerj-cs.747/fig-2

protecting against inconsistent system states. The time to finish the sub-task is therefore not the same for each virtual machine.

Using $t_f$ and $t_l$ to denote the states within the system (*i.e.*, the time when the first results and the last results are produced, respectively), if the decision mechanism only generates the outcome after all VMs have obtained the results, the execution time of each workflow sub-task is increased by $t_l - t_f$. In order to collect the results from all VMs, the time of execution of each sub-task of the workflow will increase by $t_l - t_f$. The proposed decision mechanism converts the intermediate data produced to an MD5 value for verification. Fig. 2 shows its principle.

We presume that there are sub-tasks $T_i$ and $T_j$, and the sub-task $T_j$ relies on the sub-task $T_i$ to generate intermediate data. Confidence $C$ for each intermediate data is determined by Eq. (6) using the number of identical results.

$$C = \frac{the\ number\ of\ identical\ results}{no.\ of\ vms\ in\ virtual\ cluster} \tag{6}$$

The decision model can assume five cases during the collection of outputs.

Case 1: $C = 1$ and $AEFT <$ sub-deadline; the decision module will execute sub-task $T_j$.

Case 2: $C = 1$ and $AEFT >$ sub-deadline. In this case, two actions will be taken by the decision module. First, recycling all the VMs that executing sub-task $T_i$ because the $AEFT$ is more than sub-deadline. This delay might result from some threats. Second, executing sub-task $T_j$ because the confidence is 1.

Case 3: $0.5 < C < 1$ and $AEFT <$ sub-deadline. In this case $T_j$ is executing because the majority of results are identical.

Case 4: $0.5 < C < 1$ and $AEFT >$ sub-deadline. In this case, two actions will also be taken by the decision module. First, recycling all the VMs executing sub-task $T_i$ because the $AEFT$ exceeds the sub-deadline. Second, re-executing $T_i$ because the confidence is less than one.

Case 5: $C \leq 0.5$; sub-task $T_i$ will be re-executed instantly with the available resources.

In case 2, the confidence is 1 and $AEFT$ exceeds the sub-deadline of sub-task. We assume that the increase in $AEFT$ could happen due to some accidental faults resulting in this delay.

In case 3, the confidence is less than 1 and $AEFT$ of the virtual cluster is less than the sub-deadline. Here, we assume that this might occur as a—result of accidental faults because there is no delay during the process.

In case 4, the confidence is less than 1 and the $AEFT$ of the virtual cluster is more than the sub-task sub-deadline. In this case, we assume this low confidence level is due to threat and delay. So, the significance of the sub-task re-execution is significantly high.

In case 5, the confidence is less than or equal to 0.5; thus, the current sub-task must be re-executed because of the low confidence level of the intermediate data.

Without reducing efficiency, the suggested decision mechanism will enhance the credibility of effective workflow execution.

## Fault and intrusion-tolerant workflow scheduling algorithm

To use the FITSW algorithm, first determine the sub-deadline for each sub-task using a deadline partitioning method. An elastic resource provisioning scheme is built based on these sub-deadlines, and a task-VM mapping technique is used to improve performance.

### *Deadline partitioning*

To ensure that FITSW meets the workflow deadline, we use the approach of dividing the deadline (*Cao et al., 2019*; *Wang et al., 2021*). This divides the deadline given by the user into sub-deadlines for individual scientific workflow sub-tasks. First, this approach uses Eq. (7) to measure upward rank for each subtask,

$$rank_u(t_i) = max_{t_j \in succ(t_i)} \left( \gamma_j . \frac{size(d_{i,j})}{BW} + rank_u(t_j) \right) + \omega_i^* \tag{7}$$

$\omega_i^*$ shows the time $t_i$ of the fastest VM. In Eq. (8), $\gamma_j$ is used to determine communication costs size $(d_{i,j})$/BW.

$$\gamma_j = \begin{cases} 0, & 1 - \theta^{-k_j} \\ 1 & otherwise \end{cases} \tag{8}$$

$\theta$ is a parameter that is predefined to be greater than 1. In our experiments, we set the value of $\theta$ to 1.2. $K_j$ denotes the ratio execution time $t_j$ to communication time, which can be determined by Eq. (9).

$$k_j = \frac{\omega_i^*.\mathrm{BW}}{d_{i,j}} \tag{9}$$

Then, the sub-deadline of sub-task $t_i$, *i.e.*, $subD(t_i)$ can be calculated by Eq. (10).

$$subD(t_i) = \frac{rank_u(t_{init}) - rank_u(t_i) + \omega_i^*}{rank_u(t_{init})}.D \tag{10}$$

where $t_{init}$ denotes the workflow initial sub-task and $D$ represents the deadline provided by the user.

### Dynamic task scheduling strategy based on resource circulation

The workflow environment in a traditional cloud workflow system is static and unchanged. As a result, attackers will launch backdoor attacks and leak information. If attackers gain access to VMs, they can maintain control over them for a prolonged period of time. A dynamic task scheduling approach based on resource circulation is proposed as a solution to this problem. The method recycles VMs that perform sub-tasks and creates new VMs without raising users' sub-task resource requirements. The dynamic task scheduling method is implemented using the decision process depicted in Algorithm 1.

The proposed task scheduling model can be used for VM cleaning. If viruses infect any VM during the workflow, the proposed job/task scheduling strategy would clear the affected VM. In addition, the dynamic task scheduling technique will prevent attackers from sniffing into workflow data on a regular basis. With the dynamic task scheduling approach, the proposed framework would recycle VMs that had completed subtask execution. It will also prevent attackers from taking control of a virtual machine for an extended period of time.

## EXPERIMENTS AND RESULTS

### Experimental setting

Experiments were performed using WorkflowSim (*Chen & Deelman, 2012*), an open-source cloud workflow simulation software, where the scientific workflow is represented in XML. The Montage, Epigenomics, CyberShake, Inspiral, and Sipht scientific workflows introduced by Pegasus (*Deelman et al., 2015*) were used for these experiments. The scientific workflow structures are provided in Fig. 3 while the parameters are given in Table 1. Three metrics were used to measure the performance of the algorithm:

1. Success Rate (SR) achieved after executing scientific workflows (when there are attacks): SR denotes the algorithm's intrusion tolerance.
2. Workflow Makespan: This represents the algorithm's completion time.
3. Task Completion Rate (*TCR*) (*Yao, Ding & Hao, 2017*): Calculated as shown in Eq. (11).

$$TCR = \frac{N\ baseline}{N\ workflow} \tag{11}$$

**Algorithm 1** **FITSW.**

**BEGIN**

1. **Input** $K(The\ number\ of\ VMs\ included\ in\ vrtual\ cluster)$

2. $rank_u(t_i) = \overline{\omega_i} + max_{t_j \in succ(t_i)}\left\{\frac{d_{i,j}}{BW} + rank_u(t_j)\right\}$;

//*Calculate the upword ranking of sub_task* $T_i$ *according to* (1)

3. $subD(t_i) = \frac{rank_u(t_{init}) - rank_u(t_i) + \omega_i^*}{rank_u(t_{init})}.D$;

//*Calculate the sub_deadline of sub_task* $T_i$ *according to* (10);

4. *Recieving sub_task* $T_i$ *replicas execution results from virtual clusters*;

5. $confidence = \frac{the\ number\ of\ identical\ results}{no.of\ vms\ in\ virtual\ cluster}$;

//*Calulate the confidance of intermediate data of sub_task* $T_i$ *according to* (6);

6. $AEFT(t_i) = \frac{\sum_{x=first}^{third} EFT\left(t_i^x, vm_j\right)}{no.of\ vms\ in\ virtual\ cluster}$;

//*Calculate the AEFT of virtual cluster according to* (5);

7. **if**$(confidence = 1\ and\ AEFT < sub\_deadline)$

8. *generating new VMs to executing sub_task* $T_j$;

9. **if**$(confidence = 1\ and\ AEFT > sub\_deadline)$

10. *Recycling all the VMs executing sub_task* $T_i$;

11. *generating new VMs to executing sub_task*$T_j$;

12. **if**$((0.5 < confidence < 1)\ and\ (AEFT < sub\_deadline))$

13. *generating new VMs to executing sub_task* $T_j$;

14. **if**$((0.5 < confidence < 1)\ and\ (AEFT > sub\_deadline))$

15. *Recycling all the VMs executing sub_task* $T_i$;

16. *Re_execute sub_task* $T_i$;

17. **if**$(confidence \leq 0.5)$

18. *Re_execute sub_task* $T_i$;

**END**

The HEFT workflow makespan is used as a baseline while *Nbaseline* denotes the number of sub-tasks performed at baseline time. *Nworkflow* reflects the number of workflow sub-tasks that were used in the test. The *TCR* demonstrates the algorithm's efficiency.

To evaluate FITSW, three experiments are conducted with the following aims: (1) Evaluating FITSW's *SR* with different numbers of VM's attack (2) Evaluating FITSW's *TCR*. (3) Evaluating FITSW system's efficacy by calculating completion time without considering attacks. In this experiment, we compare FITSW with ITSW (*Wang et al., 2019a*) and ITSW-RV with a random number of VMs. For the final results presentation, average values are considered. Each test is carried out twenty-five times. The processing power of each VM is produced randomly during each experiment from 1,300 *MIPS* to 2,000 *MIPS*.

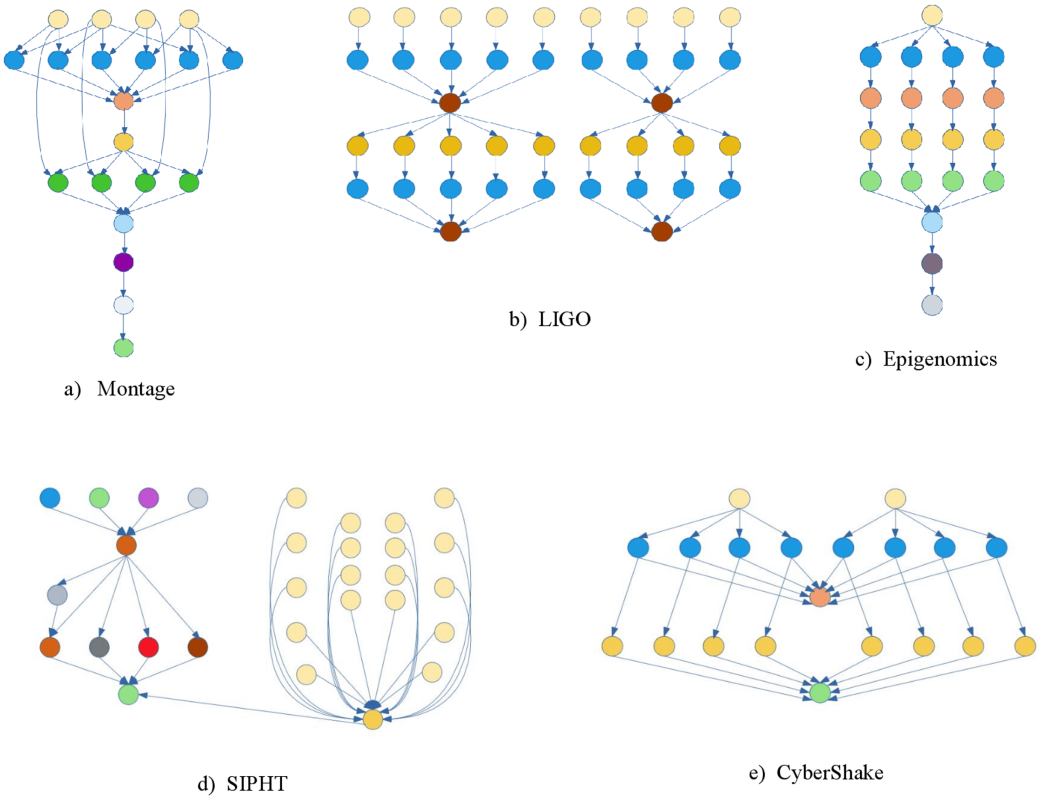


**Figure 3** Some scientific workflows structures. Full-size DOI: 10.7717/peerj-cs.747/fig-3

**Table 1** The parameters of different workflows.

| Scientific workflow | Number of sub-task | Number of edges | Average data size (MB) | Average sub-task runtime (s) |
|---|---|---|---|---|
| CyberShake_1000 | 1,000 | 3,988 | 102.29 | 22.71 |
| Epigenomics_997 | 997 | 3,228 | 388.59 | 3,858.67 |
| Montage_1000 | 1,000 | 4,485 | 3.21 | 11.36 |
| Inspiral_1000 | 1,000 | 3,248 | 8.90 | 227.25 |
| Sipht_1000 | 1,000 | 3,528 | 5.00 | 179.05 |

**Table 2** SR of FITSW, ITSW and ITSW-RV under the condition of one compromised VM.

| Algorithms | The number of available VMs | | | | | | |
|---|---|---|---|---|---|---|---|
| | 30 | 50 | 100 | 150 | 200 | 250 | 300 |
| ITSW | 0.04 | 0.08 | 0.28 | 0.6 | 0.52 | 0.68 | 0.8 |
| ITSW-RV | 0.16 | 0.16 | 0.6 | 0.52 | 0.56 | 0.72 | 0.8 |
| FITSW | 0.16 | 0.24 | 0.76 | 0.64 | 0.72 | 0.92 | 1 |

**Table 3** SR of FITSW, ITSW and ITSW-RV under the condition of two compromised VMs.

| Algorithms | The number of available VMs | | | | | | |
|---|---|---|---|---|---|---|---|
| | 30 | 50 | 100 | 150 | 200 | 250 | 300 |
| ITSW | 0.04 | 0.08 | 0.4 | 0.48 | 0.40 | 0.64 | 0.76 |
| ITSW-RV | 0.08 | 0.08 | 0.56 | 0.56 | 0.52 | 0.64 | 0.8 |
| FITSW | 0.16 | 0.24 | 0.52 | 0.58 | 0.64 | 0.8 | 0.8 |

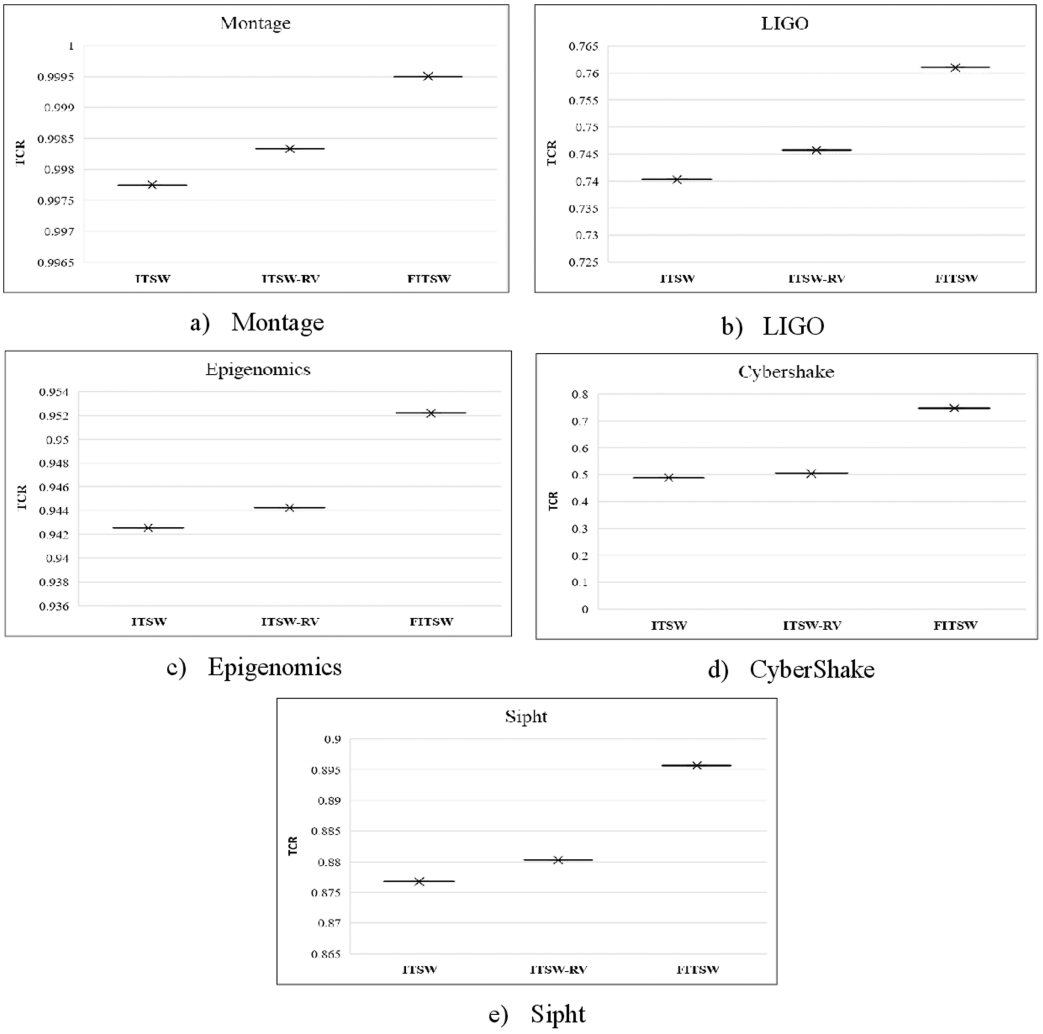


**Figure 4** (A–E) The TCR of FITSW, ITSW and ITSW-RV.
Full-size DOI: 10.7717/peerj-cs.747/fig-4

## Evaluating SR

The intrusion tolerance is quantified by *SR* in this experiment. The higher the algorithm's *SR*, the greater the intrusion tolerance for the same number of compromised VMs.

$$SR = \frac{number\ of\ simulation\ runs\ that\ successfully\ meet\ the\ deadline}{total\ number\ of\ simulation\ runs} \tag{12}$$

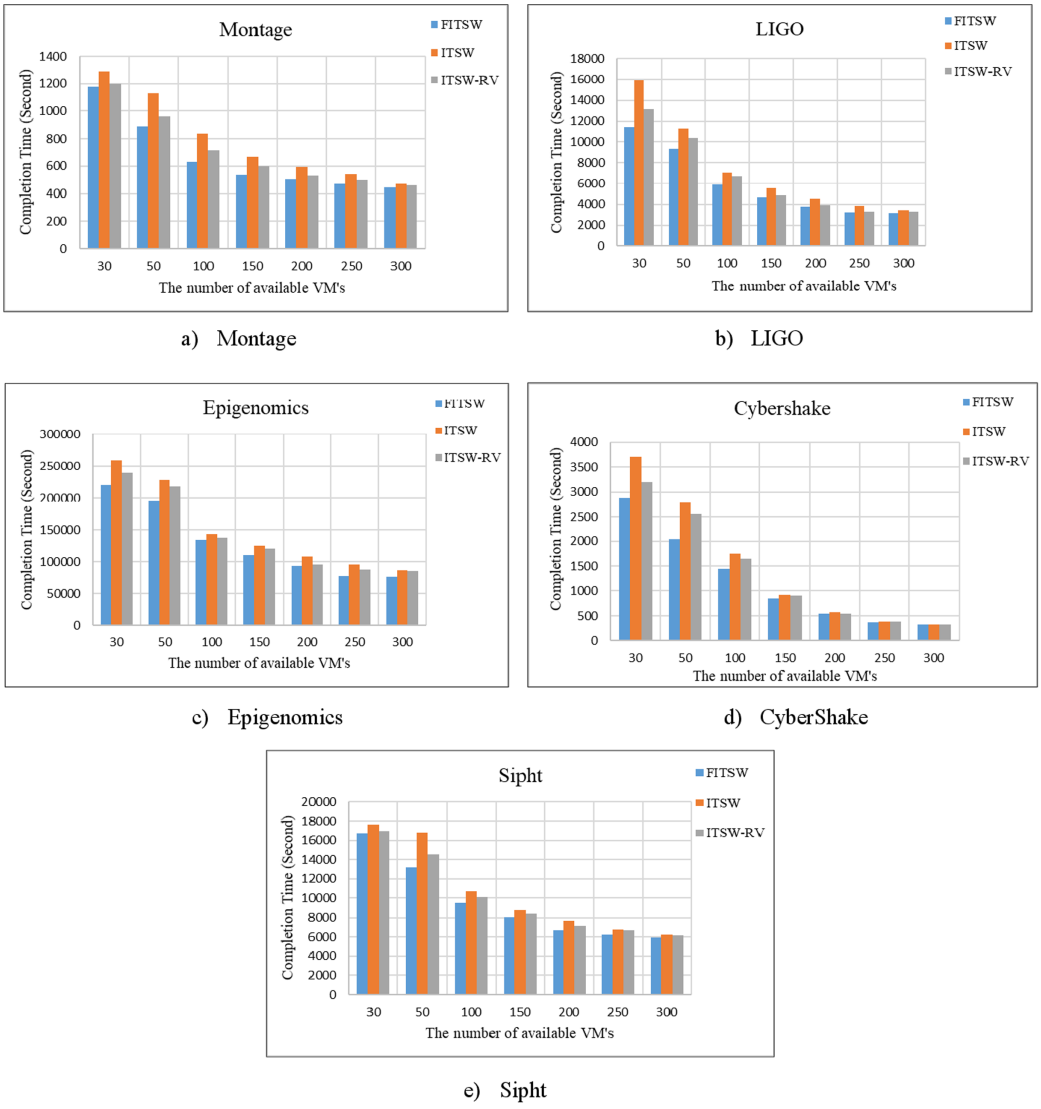


**Figure 5 (A–E) The completion time of FITSW, ITSW and ITSW-RV.**
Full-size DOI: 10.7717/peerj-cs.747/fig-5

First, we presume that the adversaries capture one VM, which generates an incorrect result for any sub-task assigned to this VM. The average *SR* of FITSW, ITSW, and ITSW-RV for different numbers of available VMs is determined for these purposes using five different scientific workflows. Tables 2 and 3 demonstrate the results.

Second, as the number of compromised VMs increases, the FITSW and ITSW *SR*s decrease dramatically. However, ITSW exhibits a more obvious downward trend than FITSW. Also, the number of available VMs influences FITSW and ITSW. This is because the higher number of VMs available, the lower the chances that two compromised VMs will be assigned to two of the three sub-task replicas. In general, FITSW is more intrusion-tolerant than ITSW.

### Evaluating TCR of FITSW

Here, we test TCR with a random number of VMs for FITSW, ITSW and ITSW-RV. TCR is defined as the relationship between tasks completed and the total number of workflow tasks checked during the soft-deadline. It reflects the effectiveness of the task performed with respect to the compared algorithms. At first, there are 100 VMs available and the HEFT makespan is the baseline. The deadline for FITSW, ITSW and ITSW-RV is (1.2*Baseline). From the result shown in Fig. 4, the TCR of ITSW is lower than that of FITSW and ITSW-RV. This results because FITSW is not based on a backup mechanism and the number of VMs available is not enough to meet the execution requirements. To support cloud-based scientific workflows, ITSW-RV can increase its resource pool while FITSW and ITSW have a fixed number of VMs.

### Efficiency assessment of FITSW system

First, without considering an attack, we checked the FITSW system's efficiency. The number of VMs that are available is used as variables. FITSW, ITSW and ITSW-RV systems workflow completion time is shown in Fig. 5. From the figure, it is obvious that the completion time of FITSW system is smaller than ITSW and ITWS-RV. This can be traced to the fact that FITSW checks the confidence value and sub-deadline for each sub-task to decide on whether to re-execute the current sub-task or execute the next. ITSW and ITSW-RV will inevitably take more time waiting for results because of its intermediate data backup mechanism for temporary workflow. The completion time of the FITSW system decreased by around 15.6% on Montage, 19% on Inspiral, 18% on CyberShake, 13% on Epigenomics, and 11% on Sipht.

## CONCLUSION

To address the security challenges of scientific workflow systems in the cloud, we propose a FITSW system. The workflow executors in this system are virtual clusters comprised of many VMs, which can improve workflow execution reliability. To detect accidental or malicious errors during the workflow scheduling process, FITSW divides the entire workflow deadline into sub-deadlines for each sub-task. The reliability of the workflow execution is further improved using a new decision mechanism to eliminate unreliable results. Since the workflow environment in a conventional cloud workflow system is static and unchanged, attackers may easily implant a backdoor attack and cause data leakage. To solve this problem, a dynamic task scheduling approach based on resource circulation is introduced. The approach disrupts the attack chain and guarantees that the task executors remain in a clean state. Performance evaluation using the WorkflowSim toolkit shows that the proposed solution achieves an improved scientific workflow intrusion tolerance. The results reveal that FITSW algorithm not only increases the success rate by about 12% but also improves task completion rate by 6.2% and reduces completion time by 15.6%, in comparison to the intrusion tolerant scientific workflow ITSW system.

## ACKNOWLEDGEMENTS

Throughout the study, the authors gratefully appreciate all those who contributed academically, technically, and administratively to this study.

## ADDITIONAL INFORMATION AND DECLARATIONS

### Funding

This work was supported by a Putra Grant, University Putra Malaysia, under Grant 95960000, and Ministry of Education (MOE) Malaysia. There was no additional external funding received for this study. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

### Grant Disclosures

The following grant information was disclosed by the authors:
Putra Grant, University Putra Malaysia: 95960000.
Ministry of Education (MOE) Malaysia.

### Competing Interests

The authors declare that they have no competing interests.

### Author Contributions

- Mazen Farid conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Rohaya Latip analyzed the data, authored or reviewed drafts of the paper, and approved the final draft.
- Masnida Hussin conceived and designed the experiments, authored or reviewed drafts of the paper, and approved the final draft.
- Nor Asilah Wati Abdul Hamid analyzed the data, authored or reviewed drafts of the paper, and approved the final draft.

### Data Availability

The following information was supplied regarding data availability:

The different scientific workflows that we used in our experiments are available as Supplemental Files.

### Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.747#supplemental-information.

## REFERENCES

**Ala'Anzy M, Othman M. 2019.** Load balancing and server consolidation in cloud computing environments: a meta-study. *IEEE Access* **7**:141868–141887 DOI 10.1109/ACCESS.2019.2944420.

**Aslam S, Islam S, Khan A, Ahmed M, Akhundzada A, Khan MK. 2017.** Information collection centric techniques for cloud resource management: taxonomy, analysis and challenges. *Journal of Network and Computer Applications* **100(1)**:80–94 DOI 10.1016/j.jnca.2017.10.021.

**Atya AOF, Qian Z, Krishnamurthy SV, La Porta T, McDaniel P, Marvel L. 2017.** Malicious co-residency on the cloud: attacks and defense. In: *Proceedings of the IEEE INFOCOM*. Piscataway: IEEE.

**Bhattarai S, Rook S, Ge L, Wei S, Yu W, Fu X. 2014.** On simulation studies of cyber attacks against LTE networks. In: *Proceedings of the International Conference on Computer Communications and Networks, ICCCN*.

**Bhattarai S, Wei S, Rook S, Yu W, Erbacher RF, Cam H. 2015.** On simulation studies of jamming threats against LTE networks. In: *2015 International Conference on Computing, Networking and Communications, ICNC 2015*. 99–103.

**Cao S, Deng K, Ren K, Li X, Nie T, Song J. 2019.** A deadline-constrained scheduling algorithm for scientific workflows in clouds. In: *Proceedings of the 21st IEEE International Conference on High Performance Computing and Communications, 17th IEEE International Conference on Smart City and 5th IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS*. Piscataway: IEEE, 98–105.

**Chang V, Wills G. 2016.** A model to compare cloud and non-cloud storage of Big Data. *Future Generation Computer Systems* **57(4)**:56–76 DOI 10.1016/j.future.2015.10.003.

**Chen H, Zhu X, Qiu D, Liu L, Du Z. 2017.** Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds. *IEEE Transactions on Parallel and Distributed Systems* **28(9)**:2674–2688 DOI 10.1109/TPDS.2017.2678507.

**Chen W, Deelman E. 2012.** WorkflowSim: a toolkit for simulating scientific workflows in distributed environments. In: *2012 IEEE 8th International Conference on E-Science, e-Science 2012*. Piscataway: IEEE.

**Deelman E, Vahi K, Juve G, Rynge M, Callaghan S, Maechling PJ, Mayani R, Chen W, Ferreira Da Silva R, Livny M, Wenger K. 2015.** Pegasus, a workflow management system for science automation. *Future Generation Computer Systems* **46(3)**:17–35 DOI 10.1016/j.future.2014.10.008.

**Ding Y, Yao G, Hao K. 2017.** Fault-tolerant elastic scheduling algorithm for workflow in Cloud systems. *Information Sciences* **393(8)**:47–65 DOI 10.1016/j.ins.2017.01.035.

**Farid M, Latip R, Hussin M, Abdul Hamid NAW. 2020.** Scheduling scientific workflow using multi-objective algorithm with fuzzy resource utilization in multi-cloud environment. *IEEE Access* **8**:24309–24322 DOI 10.1109/ACCESS.2020.2970475.

**Ferdaus MH, Murshed M, Calheiros RN, Buyya R. 2017.** An algorithm for network and data-aware placement of multi-tier applications in cloud data centers. *Journal of Network and Computer Applications* **98(13)**:65–83 DOI 10.1016/j.jnca.2017.09.009.

**Ghazouani S, Slimani Y. 2017.** A survey on cloud service description. *Journal of Network and Computer Applications* **91(May)**:61–74 DOI 10.1016/j.jnca.2017.04.013.

**Gill SS, Buyya R. 2018.** SECURE: self-protection approach in cloud resource management. *IEEE Cloud Computing* **5(1)**:60–72 DOI 10.1109/MCC.2018.011791715.

**Gupta P, Gupta PK. 2020.** Trust & fault in multi layered cloud computing architecture. In: *Trust & Fault in Multi Layered Cloud Computing Architecture*.

**Javadi B, Kondo D, Vincent JM, Anderson DP. 2011.** Discovering statistical models of availability in large distributed systems: an empirical study of SETI@home. *IEEE Transactions on Parallel and Distributed Systems* **22(11)**:1896–1903 DOI 10.1109/TPDS.2011.50.

**Jhawar R, Piuri V. 2012.** Fault tolerance management in cloud computing: a system-level perspective. In: *Proceedings of the 2012 IEEE 1st AESS European Conference on Satellite Telecommunications, ESTEL*. Piscataway: IEEE, 1–10.

**Jiang H, Haihong E, Song M. 2017.** Dynamic scheduling of workflow for makespan and robustness improvement in the IaaS cloud. *IEICE Transactions on Information and Systems* **E100D(4)**:813–821 DOI 10.1587/transinf.2016EDP7346.

**Lee YC, Han H, Zomaya AY, Yousif M. 2015.** Resource-efficient workflow scheduling in clouds. *Knowledge-Based Systems* **80(2)**:153–162 DOI 10.1016/j.knosys.2015.02.012.

**Li Z, Ge J, Yang H, Huang L, Hu HHH, Hu HHH, Luo B. 2016.** A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. *Future Generation Computer Systems* **65(3)**:140–152 DOI 10.1016/j.future.2015.12.014.

**Lin C, Lu S, Fei X, Chebotko A, Pai D, Lai Z, Fotouhi F, Hua J. 2009.** A reference architecture for scientific workflow management systems and the VIEW SOA solution. *IEEE Transactions on Services Computing* **2(1)**:79–92 DOI 10.1109/TSC.2009.4.

**Liu J, Wang S, Zhou A, Kumar SAP, Yang F, Buyya R. 2018.** Using proactive fault-tolerance approach to enhance cloud service reliability. *IEEE Transactions on Cloud Computing* **6(4)**:1191–1202 DOI 10.1109/TCC.2016.2567392.

**Liu W, Peng S, Du W, Wang W, Zeng GS. 2014.** Security-aware intermediate data placement strategy in scientific cloud workflows. *Knowledge and Information Systems* **41(2)**:423–447 DOI 10.1007/s10115-014-0755-x.

**Narayana KS, Pasupuleti SK. 2018.** *Trusted model for virtual machine security in cloud computing.* Vol. 710. Singapore: Springer.

**Nepal S, Sinnott RO, Friedrich C, Wise C, Chen S, Kanwal S, Yao J, Lonie A. 2017.** TruXy: trusted storage cloud for scientific workflows. *IEEE Transactions on Cloud Computing* **5(3)**:428–442 DOI 10.1109/TCC.2015.2489638.

**Pezoa JE, Dhakal S, Hayat MM. 2010.** Maximizing service reliability in distributed computing systems with random node failures: theory and implementation. *IEEE Transactions on Parallel and Distributed Systems* **21(10)**:1531–1544 DOI 10.1109/TPDS.2010.34.

**Poola D, Ramamohanarao K, Buyya R. 2016.** Enhancing reliability of workflow execution using task replication and spot instances. *ACM Transactions on Autonomous and Adaptive Systems* **10(4)**:1–21 DOI 10.1145/2815624.

**Qin X, Jiang H. 2006.** A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems. *Parallel Computing* **32(5–6)**:331–356 DOI 10.1016/j.parco.2006.06.006.

**Qiu W, Zheng Z, Wang X, Yang X, Lyu MR. 2014.** Reliability-based design optimization for cloud migration. *IEEE Transactions on Services Computing* **7(2)**:223–236 DOI 10.1109/TSC.2013.38.

**Rao J, Wei Y, Gong J, Xu CZ. 2013.** QoS guarantees and service differentiation for dynamic cloud applications. *IEEE Transactions on Network and Service Management* **10(1)**:43–55 DOI 10.1109/TNSM.2012.091012.120238.

**Rodriguez MA, Buyya R. 2017.** A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. *Concurrency Computation* **29(8)**:1–23 DOI 10.1002/cpe.4041.

**Sangaiah AK, Suraki MY, Sadeghilalimi M, Bozorgi SM, Hosseinabadi AAR, Wang J. 2019.** A new meta-heuristic algorithm for solving the flexible dynamic job-shop problem with parallel machines. *Symmetry* **11(2)**:1–17 DOI 10.3390/sym11020165.

**Sun G, Liao D, Zhao D, Xu Z, Yu H. 2018.** Live migration for multiple correlated virtual machines in cloud-based data centers. *IEEE Transactions on Services Computing* **11(2)**:279–291 DOI 10.1109/TSC.2015.2477825.

**Szefer J, Keller E, Lee RB, Rexford J. 2011.** Eliminating the hypervisor attack surface for a more secure cloud categories and subject descriptors. In: *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*. 401–412.

**Teylo L, de Paula U, Frota Y, de Oliveira D, Drummond LMMA. 2017.** A hybrid evolutionary algorithm for task scheduling and data assignment of data-intensive scientific workflows on clouds. *Future Generation Computer Systems* **76(1)**:1–17 DOI 10.1016/j.future.2017.05.017.

**Topcuoglu H, Hariri S, Wu MY. 2002.** Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems* **13(3)**:260–274 DOI 10.1109/71.993206.

**Wang Y-W, Wu J-X, Guo Y-F, Hu H-C, Liu W-Y, Cheng G-Z. 2018.** Scientific workflow execution system based on mimic defense in the cloud environment. *Frontiers of Information Technology and Electronic Engineering* **19(12)**:1522–1536 DOI 10.1631/FITEE.1800621.

**Wang Y, Guo Y, Guo Z, Baker T, Liu W. 2020.** CLOSURE: a cloud scientific workflow scheduling algorithm based on attack-defense game model. *Future Generation Computer Systems* **111(3)**:460–474 DOI 10.1016/j.future.2019.11.003.

**Wang Y, Guo Y, Guo Z, Liu W, Yang C. 2019a.** Protecting scientific workflows in clouds with an intrusion tolerant system. *IET Information Security* **14(2)**:157–165 DOI 10.1049/iet-ifs.2018.5279.

**Wang Y, Guo Y, Guo Z, Liu W, Yang C. 2019b.** Securing the intermediate data of scientific workflows in clouds with ACISO. *IEEE Access* **7**:126603–126617 DOI 10.1109/ACCESS.2019.2938823.

**Wang Y, Guo Y, Wang W, Liang H, Huo S. 2021.** INHIBITOR: an intrusion tolerant scheduling algorithm in cloud-based scientific workflow system. *Future Generation Computer Systems* **114(9)**:272–284 DOI 10.1016/j.future.2020.08.004.

**Wang Z, Wu J, Guo Z, Cheng G, Hu H. 2016.** Secure virtual network embedding to mitigate the risk of covert channel attacks. In: *Proceedings of the IEEE INFOCOM*. Piscataway: IEEE, 144–145.

**Wu J, Lei Z, Chen S, Shen W. 2017.** An access control model for preventing virtual machine escape attack. *Future Internet* **9(2)**:20 DOI 10.3390/fi9020020.

**Yao G, Ding Y, Hao K. 2017.** Using imbalance characteristic for fault-tolerant workflow scheduling in cloud systems. *IEEE Transactions on Parallel and Distributed Systems* **28(12)**:3671–3683 DOI 10.1109/TPDS.2017.2687923.

**Yao G, Ding Y, Ren L, Hao K, Chen L. 2016.** An immune system-inspired rescheduling algorithm for workflow in cloud systems. *Knowledge-Based Systems* **99(6)**:39–50 DOI 10.1016/j.knosys.2016.01.037.

**Yu W, Liang F, He X, Hatcher WG, Lu C, Lin J, Yang X. 2017.** A survey on the edge computing for the internet of things. *IEEE Access* **6**:6900–6919 DOI 10.1109/ACCESS.2017.2778504.

**Yuan D, Yang Y, Liu X, And GZ, Chen J. 2010.** A data dependency based strategy for intermediate data storage in scientific cloud workflow systems. *Concurrency Computation Practice and Experience* **22(6)**:685–701 DOI 10.1002/cpe.1636.

**Zhang Y, Juels A, Reiter MK, Ristenpart T. 2014.** Cross-tenant side-channel attacks in PaaS clouds. In: *Proceedings of the ACM Conference on Computer and Communications Security*. New York: ACM, 990–1003.

**Zhao Y, Li Y, Raicu I, Lin C, Tian W, Xue R. 2014.** Migrating scientific workflow management systems from the grid to the cloud. *Cloud Computing for Data-Intensive Applications* 231–256 DOI 10.1007/978-1-4939-1905-5.

**Zheng Z, Zhou TC, Lyu MR, King I. 2012.** Component ranking for fault-tolerant cloud applications. *IEEE Transactions on Services Computing* **5(4)**:540–550 DOI 10.1109/TSC.2011.42.

**Zhou AC, He B, Liu C. 2016.** Monetary cost optimizations for hosting workflow-as-a-service in IaaS clouds. *IEEE Transactions on Cloud Computing* **4(1)**:34–48 DOI 10.1109/TCC.2015.2404807.