Fault-Tolerant Scheduling for Scientific Workflow with Task Replication Method in Cloud

Zhongjin Li[1], Jiacheng Yu[1], Haiyang Hu[1], Jie Chen[1], Hua Hu[1], Jidong Ge[2] and Victor Chang[3]
[1]*School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China*
[2]*State Key Laboratory for Novel Software Technology, Software Institute, Nanjing University, China*
[3]*International Business School Suzhou, Xi'an Jiaotong Liverpool University, Suzhou, China*

Keywords: Cloud Computing, Fault-tolerant, Scientific Workflow Scheduling.

Abstract: Cloud computing has become a revolutionary paradigm by provisioning on-demand and low cost computing resources for customers. As a result, scientific workflow, which is the big data application, is increasingly prone to adopt cloud computing resources. However, internal failure (host fault) is inevitable in such large distributed computing environment. It is also well studied that cloud data center will experience malicious attacks frequently. Hence, external failure (failure by malicious attack) should also be considered when executing scientific workflows in cloud. In this paper, a fault-tolerant scheduling (FTS) algorithm is proposed for scientific workflow in cloud computing environment, the aim of which is to minimize the workflow cost with the deadline constraint even in the presence of internal and external failures. The FTS algorithm, based on tasks replication method, is one of the widely used fault tolerant mechanisms. The experimental results in terms of real-world scientific workflow applications demonstrate the effectiveness and practicality of our proposed algorithm.

1 INTRODUCTION

Cloud computing is the popular and promising computing platforms for users or customers, and its on-demand computational resources can be obtained easily in the form of virtual machine (VM) (Foster et al., 2008; Mell and Grance, 2009; Sun et al., 2016). Workflow is common formed by a number of tasks and the control structures, which typically modeled as a directed acyclic graph (DAG) (Kyriazis et al., 2008). It is used to model scientific computing applications, such as physics, bioinformatics, astronomy, numerical weather forecast and so on (Li et al., 2017). With the growth complexity of these applications, scientific workflows are become big data applications and require large-scale infrastructures to conduct in a reasonable time (Li et al., 2016; Rodriguez and Buyya, 2014]. Accordingly, scientific workflows are prone to exploit the cloud computing resources (Kashlev and Lu, 2014; Zhao et al., 2011).

Although executing scientific workflows on cloud platform bring many advantages, cloud computing, similar to other distributed computing system, is also easily to emerge resource failures. Some of them result from internal failure (i.e., host fault) (Jeannot et al., 2012; Zhu et al., 2016; Yao et al., 2017, Qiu et al., 2017). According to the report that a system consists of 10 thousand physical servers, one will fail once a day (Dean, 2009). Moreover, about 1-5 percentage of disk drives die and hosts crash at least twice with roughly 2-4 percentage every year (Zhu et al., 2016). Thus, the workflow application is likely to delay even if only one server fails during the task executing process. Moreover, various security threats (such as spoofing and alteration) are the great concern for cloud users and providers (Zeng et al., 2015; Li et al., 2016; Chen et al., 2017). For example, alteration is one of the malicious attacks that can lead to serious task faults by changing the execution data. Hence, external failure (i.e., failure by malicious attack) should also be considered when executing scientific workflow applications. Fortunately, integrity service, a security check method, can be utilized to ensure that no one modify or tamper with the data without being detected during the process of task executing (Xie and Qin, 2006, 2008). Hence, it is

Li, Z., Yu, J., Hu, H., Chen, J., Hu, H., Ge, J. and Chang, V.
Fault-Tolerant Scheduling for Scientific Workflow with Task Replication Method in Cloud.
DOI: 10.5220/0006687300950104
In *Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS 2018)*, pages 95-104
ISBN: 978-989-758-296-7

also necessary to deploy security service to check the integrity of running data for workflow tasks.

The task resubmission and replication methods are two extensively utilized fault tolerance methods (Chen et al., 2016; Vinay and Dilip Kumar, 2017). As for the resubmission, it resubmits a task execution after a failure happens. The resubmission mechanism is generally used during the course of task execution and can enhance the resource utilization of computing system. Nevertheless, resubmission method will result in much late finish time for tasks and may fail to meet the deadline constraint of workflow (Vinay and Dilip Kumar, 2017). Alternatively, tasks can also be duplicated to avoid failures, and the replications of a task can be executed simultaneously (Chen et al., 2016). The replication is also realized in a primary-backup mode, where the backup starts executing when the primary fails (Ghosh et al., 1997; Manimaran and Murthy, 1998; Zhu et al., 2011, 2016; Sun et al., 2017). Therefore, the replication method is applicable for the task scheduling phase and is good for saving execution time of task.

In the cloud computing environment, task failures may result from internal failure or external failures. Moreover, workflow scheduling in cloud usually takes the deadline into consideration. So, resubmission method is not applicable for cloud workflow scheduling. In this paper, we propose a fault-tolerant scheduling (FTS) algorithm for scientific workflow in cloud computing environment, the aim of which is to optimize the workflow cost with the deadline constraint even in the presence of various failures. The task replication scheme is integrated into the FTS algorithm, and the number of replications depends on the internal and external failures probabilities. The experimental results, on the basis of real-world scientific workflow applications, demonstrate the effectiveness and practicality of our proposed algorithm. The main contributions of this work are given as follows:

- We propose a fault-tolerant scheduling algorithm for scientific workflow in cloud to optimize the workflow execution cost while meeting the deadline constraint.
- The proposed FTS algorithm, which is based on task replication, can ensure the successful execution of task in the presence of internal failure (i.e., host failure) or external failure (i.e., failure by malicious attack).
- In terms of real-world scientific workflow applications, our experiments demonstrate the effectiveness and practicality of our proposed FTS algorithm.

The remainder of this paper is organized as follows. Section 2 summarizes the related work. Section 3 describes the models and problem formulation. Section 4 introduces the algorithm implementation. Section 5 analyses the experimental results. Finally, the conclusions and future work are given in Section 6.

2 RELATED WORK

The problems of workflow scheduling in cloud have been well studied recently. Zhao et al. (2011) present the key challenges and research opportunities in running scientific workflow on cloud. Then, a cloud scientific workflow management system is proposed, which integrates Swift system with the OpenNebula cloud computing platform (Zhao et al., 2012). In commercial multi-cloud environment, Fard et al. (2013) introduce a pricing and truthful model for workflow scheduling to minimize the workflow makespan and monetary cost simultaneously. Rodriguez and Buyya (2014) propose a scientific workflow scheduling mechanism according to Infrastructure as a Service (IaaS) that optimizes the entire workflow scheduling cost with the deadline constraint. Li et al. (2017) present a cost and energy aware workflow scheduling algorithm, which is based on four optimization steps, to minimize the workflow cost and reduce the energy consumption under the constraint of workflow deadline. Furthermore, many studies have concentrated on multiple objective workflow scheduling problems in cloud. Durillo et al. (2012) propose a multi-objective list scheduling heuristic for workflow in cloud computing environment. As an alternative, Zhu et al. (2016) develop an evolutionary multi-objective optimization (EMO) algorithm for scientific workflow scheduling in the same scenario. However, the aforementioned workflow management system, single objective and multi-objective scheduling algorithms neglect the tasks failure problem that will influence the quality of service (QoS) of workflow.

Since the occurrences of internal faults are usually unpredictable in computer systems, fault tolerance must be considered when devising workflow scheduling algorithms. Ghosh et al. (1997) provide a fault-tolerance technique in dynamic systems that can help system designers determine how many processors should be needed. Manimaran and Murthy (1998) propose a fault-tolerant algorithm to dynamically schedule real-time tasks in the multiprocessor system. Zhu et al. (2011) present a fault-tolerant scheduling algorithm that can

Table 1: Notations.

Symbol	Semantics
t_i	Task t_i of workflow
$D(t_i)$	The size of input data of task t_i
$W(t_i)$	Workload of task t_i
$pre(t_i)$	Predecessor set of task t_i
$succ(t_i)$	Successor set of task t_i
n	The number of tasks of workflow
$VM(k)$	The $k$thVM type
$P(k)$	Processing capacity of $VM(k)$
$C(k)$	The cost per unit time of $VM(k)$
B	The bandwidth between VMs
λ	Failure coefficient
β	Weight parameter
$P_{fault}(t_i)$	Fault probability of task t_i
$n_{copy}(t_i)$	The number of copies of task t_i
$T_{trans}(t_i)$	Transmission time of task t_i
$T_{exec}(t_i, VM(k))$	Execution time of task t_i
$T_{start}(t_i)$	Start time of task t_i
$T_{end}(t_i)$	End time of task t_i
$T_{rent}(t_i, VM(k)$	VM rent time of task t_i on $VM(k)$
$cost(t_i, VM(k))$	VM rent cost of task t_i on $VM(k)$
$cost$	The cost of workflow
$makespan$	The makespan of workflow

tolerate one node failures for real-time tasks in heterogeneous cluster environment. However, these fault-tolerant scheduling algorithms cannot be directly applied to cloud computing environment or workflow scheduling problem. In (Plankensteiner and Prodan, 2012), a resubmission heuristic strategy is proposed to support fault tolerant execution of scientific workflows. Wang et al. (2015) present a fault-tolerant mechanism which extends the primary-backup model to cloud computing system. Chen et al. (2016) propose three clustering strategies of fault tolerant to improve the QoS of workflow. Zhu et al. (2016) construct a real-time workflow fault-tolerant model that extends the traditional primary-backup model based on many cloud computing characteristics, and the task allocation and message transmission mechanism are developed to ensure task faults can be done in the process of workflow executing. However, the fault-tolerant methods mentioned above only consider the internal faults, but they ignore the external faults.

The security problem in workflow scheduling has been studied to deal with external malicious attacks in cloud. Chen et al. (2017) investigate the problems of workflow scheduling with security-sensitive intermediate data. Li et al. (2016) propose a security and cost aware scheduling algorithm for scientific workflow, the aim of which is to optimize the workflow cost under the deadline and risk rate constraints. Zeng et al. (2015) propose a security-aware and budget-aware (SABA) workflow scheduling scheme to minimize makespan within both the security and budget constraints. However, existing algorithms only consider the security constraints for workflow scheduling and are incapable to solve the failure problem by malicious attack.

Unlike the aforementioned approaches, in this study, we task both internal and external failures into count simultaneously and propose a fault-tolerant scheduling (FTS) algorithm for scientific workflow in cloud computing environment. The FTS algorithm is based on tasks replication method (one of the widely used fault tolerant mechanisms), and the aim of which is to minimize the workflow cost with the deadline constraint even in the presence of various failures. The experimental results in terms of four real-world scientific workflow applications demonstrate the effectiveness and practicality of our proposed algorithm.

3 MODELS AND PROBLEM FORMULATION

In this section, first we describe some models used in this paper, including workflow model, cloud model and fault model. Then, the problem formulation of fault-tolerant scientific workflow scheduling is introduced. The major notations and their semantics in this paper are summarized in Table 1.

3.1 Workflow Model

The model of workflow is usually represented by the DAG (directed acyclic graph) model, that is $WF = (T, E)$, where $T = \{t_0, t_1, \dots, t_i, \dots, t_{n-1}\}$ is the workflow tasks set, $E = \{(t_i, t_j) | t_i, t_j \in T\}$ is the set of edges between tasks. Let $pre(t_i)$ and $succ(t_i)$ represent the set of predecessor and set of successor of task t_i respectively. Then, suppose a DAG has exactly one entry task and one exit task, and a task is called entry task t_{entry}, if and only if $pre(t_{entry}) = \emptyset$; a task is called exit task t_{exit}, if and only if $succ(t_{exit}) = \emptyset$. Moreover, symbol $W(t_i)$ is the workload of task t_i, which is quantified in unit of compute unit, and $D(t_i)$ represents the size of the input data of tasks t_i. In addition, each workflow has a deadline $T_{deadline}$ which is defined as the constraint of execution time.

Table 2: m4 series of VMs in Amazon EC2.

VM number	VM type	Compute Unit	Cost per Hour ($)
1	m4.large	2	0.1
2	m4.xlarge	4	0.2
3	m4.2large	8	0.4
4	m4.4large	16	0.8
5	m4.10large	40	2
6	m4.16large	64	3.2

3.2 Cloud Model

Through virtualization technology, each server in cloud systems can be virtualized to a set of heterogeneous VMs. Hence, the VM is the basic processor unit in cloud instead of server. Suppose the cloud systems offer a set of VM resources in the form of $VM = \{VM(1), \dots, VM(k), \dots, VM(K)\}$ to users in the pay-per-use model. For example, Amazon EC2 provides six types of m4 series VMs which is shown in Table 2 (Amazon EC2, 2017). Specially, a VM instance $VM(k)$ is mainly specified by processing capacity $P(k)$ (in compute unit) and cost per hour $C(k)$. Without loss of the generality, Amazon EC2 charge users by the hourly-based pricing model that means users have to pay for the whole leased hour even if the VM leased just one minute (Rodriguez and Buyya, 2014; Li et al., 2017). Empowered by the virtualization technology, an infinite amount of VMs can be accessed in cloud computing platform, and so users can rent the arbitrary number of VMs. Moreover, all VMs located in the one cloud data center so that the bandwidth between VMs is supposed to be equal (Li et al., 2016; Yao et al., 2017).

3.3 Fault Model

We take the internal and external faults simultaneously into account in cloud workflow scheduling problem. As for internal failure, host failure is focused, which can bring about failures including VMs and workflow tasks. So, a fault-detection mechanism is used to detect host failure (Ghosh et al., 1997; Manimaran and Murthy, 1998). Furthermore, failures on hosts may be transient or permanent, independent, which means that a fault occurred on one host will not affect other hosts. Since the probability that two hosts fail simultaneously is small, we assume that at most one host fails at a time (Zhu et al., 2011, 2016).

Security threats (Snooping, spoofing and alteration) are a big concern in cloud computing system. As far as we known, snooping and spoofing attacks only incur significant data losses of workflow. However, only alteration is an unauthorized attack that can lead to invalid tasks execution, which is termed the external failure. Then, we can apply integrity service to check whether the task executing successfully (Xie and Qin, 2006; Li et al., 2016). There are many hash functions for integrity services such as TIGER, RIFDMD-160, SHA-1, RIFDMD-128, MD5, etc (Xie and Qin, 2006, Li et al., 2016; Chen et al., 2017). Each hash function is assigned a security level in the range 0 to 1. However, adding the security services to applications inevitably produces time overhead, which will increase the makespan and cost of applications. Among the above hash functions, the TIGER method, with the highest security level, has the most security overhead. Moreover, the time overhead of security service is in direct proportion to the size of data (Li et al., 2016; Chen et al., 2017). In order to check the executing data whether is altered by malicious users during the task executing, we use the TIGER method as the integrity service. Then, the time of task t_i using the TIGER method to check the execution data is computed by Eq. (1).

$$T_{secu}(t_i) = \beta \cdot D(t_i) \quad (1)$$

where β is the weight parameter of TIGER security service and $D(t_i)$ is the size of execution data of task t_i. As for scientific workflow, e.g., NCFS workflow, the size of input data may range from 0.5GB to 8.7GB (Zeng et al., 2015). Hence, the security time overhead cannot be overlooked when devising the workflow scheduling algorithm.

3.4 Problem Formulation

The aim of this paper is to minimize the workflow cost with the deadline constraint even in the presence of failures. Then, an efficient scheduling scheme of mapping workflow tasks onto VMs should be found. Then, Let the start time and end time of a task t_i as $T_{start}(t_i)$ and $T_{end}(t_i)$, and the start time of t_i is represented by Eq. (2).

$$T_{start}(t_i) = \max_{t_j \in pre(t_i)}\{T_{end}(t_j)\} \quad (2)$$

Note that if $t_i = t_{entry}$, then $T_{start}(t_i) = 0$. A task t_i can start its execution if and only if it receives input data from all its predecessors. Then, the transmission time is computed by

$$T_{trans}(t_i) = D(t_i)/B \quad (3)$$

where B is the bandwidth between two VMs in cloud computing platform. Then, the task t_i begins to execute, and the execution time is given by

$$T_{exec}(t_i, VM(k)) = W(t_i)/P(k) \quad (4)$$

Thus, the end time of task t_i is computed as follows.

$$T_{end}(t_i) = T_{start}(t_i) + T_{trans}(t_i) + T_{exec}(t_i, VM(k)) + T_{secu}(t_i) \quad (5)$$

We know that the end time of task t_{exit} is the makespan of workflow, then

$$makespan = T_{end}(t_{exit}) \quad (6)$$

Based on the above definitions, the VM rent time of task t_i executed on $VM(k)$ is given by

$$T_{rent}(t_i, VM(k)) = T_{trans}(t_i) + T_{exec}(t_i, VM(k)) + T_{secu}(t_i) \quad (7)$$

However, Amazon EC2 typically charges the users by an hourly-based pricing model. Then, the cost of one copy executed on $VM(k)$ is represented as follows.

$$cost(t_i, VM(k)) = \lceil T_{rent}(t_i, VM(k)) \rceil \cdot C(k) \quad (8)$$

In the cloud computing environment, task failures are inevitable, and we use the replication method to ensure the fault-tolerant. Hence, suppose $n_{copy}(t_i)$ is the number of replications of task t_i. Then, the cost of task t_i is calculated by

$$cost(t_i) = n_{copy}(t_i) \cdot cost(t_i, VM(k)) \quad (9)$$

Thus, he cost of workflow can be computed by

$$cost = \sum_{t_i \in T} cost(t_i) \quad (10)$$

Finally, the workflow scheduling problem can be formally defined as follows: find a schedule scheme to minimize $cost$, and the $makespan$ is equal or less than $T_{deadline}$, which is described as follows.

$$\text{Minimize: } cost \quad (11)$$

$$\text{Subject to: } makespan \leq T_{deadline} \quad (12)$$

4 ALGORITHM IMPLEMENTATION

In this section, we propose a fault-tolerant scheduling algorithm for scientific workflow to address the faults happened in internal and external cloud computing environment. The FTS algorithm is capable of reducing workflow cost while meeting the deadline. In this section, we present the implementation of proposed algorithms in detail as follows.

With the aim of meeting the deadline constraint of workflow, we first introduce a concept of sub-makespan. The sub-makespan stands for the assigned execution time of a task, which is similar to the makespan of the workflow. Obviously, if the execution time of each task is no more than its sub-makespan, then the makespan of workflow will not exceed the deadline.

First, we map each task to the maximum compute unit $VM(K)$. Then, the minimum execution time of task is calculated by

$$T_{exec}(t_i, VM(K)) = W(t_i)/P(K) \quad (13)$$

In this case, we can derive the minimum makespan of workflow $makespan^{min}$ when all tasks of workflow execute on $VM(K)$. Without loss of generality, we assume that the specified deadline $T_{deadline}$ will no less than the minimum makespan, that is $T_{deadline} \geq makespan^{\min}$. We define the sub-makespan of task t_i as $T_{subm}(t_i)$ which is represented as follows (Li et al., 2017).

$$T_{subm}(t_i) = (T_{trans}(t_i) + T_{exec}(t_i, VM(K)) + T_{secu}(t_i)) \cdot T_{deadline}/makespan^{\min} \quad (14)$$

FTS Algorithm

BEGIN
01. **for** each task $t_i \in T$
02. Calculate the minimum $T_{exec}(t_i, VM(K))$;
03. **end for**
04. Calculate the minimum makespan $makespan^{\min}$;
05. **for** each task $t_i \in T$
06. Calculate the sub-makespan based on Eq. (14);
07. Find the feasible set $VM_{feasible}(t_i)$;
08. **for** each $VM(k) \in VM_{feasible}(t_i)$
09. Find an optimal $VM_{opt}(k)$ that satisfy Eq. (20);
10. **end for**
11. **end for**
12. Compute the $makespan$ according to Eq. (6);
13. Compute the $cost$ according to Eq. (10);
END

Figure 1: The pseudo code of FTS algorithm.

Workflow executing in cloud computing environment is not risk-free. As for external failure, the failure occurrence is approximated by a Poisson distribution (Qiu et al., 2017). Then, the fault probability of task t_i executed on cloud computing platform is modeled by an exponential distribution given as follows (Fard et al., 2012).

$$P_{fault}(t_i) = 1 - \exp(-\lambda \cdot T_{rent}(t_i, VM(k))) \quad (15)$$

where λ is the failure coefficient of cloud computing environment.

To guarantee the successful execution of all tasks, we utilize the replication method to duplicate multiple copies of tasks to execute simultaneously. Then, we assume that parameter ε is a small positive integer that is approximate to zero. Then, we have

$$(P_{fault}(t_i))^{n_{copy}^{ext}(t_i)} \leq \varepsilon \quad (16)$$

where $n_{copy}^{ex}(t_i)$ is number of task t_i copies which is used to against the external failure. Also the task may be fault due to internal failure, i.e. host or server fault. Since the probability that two hosts fail simultaneously is small, hence suppose at most one host fails at a time. Then, we add another one copy to solve this problem. Thus, the total copies of task t_i calculated by Eq. (17).

$$n_{copy}(t_i) = n_{copy}^{ex}(t_i) + 1 \quad (17)$$

Eq. (17) can be explained as follows: cloud computing environment may suffer from internal and external failures, and a task replication method is used to ensure the task successful execution. First, $n_{copy}^{ex}(t_i)$ copies of tasks are aim at the external malicious attack. In additional, one extra copy is to guarantee the internal host fault. Hence, $n_{copy}(t_i)$ copies of tasks execute on VMs simultaneously with one copy of task will be successful at least. Moreover, note that if $n_{copy}(t_i) = 2$, then our multiple replication method is similar to well-known primary-backup model (Ghosh et al., 1997; Manimaran and Murthy, 1998; Zhu et al., 2016).

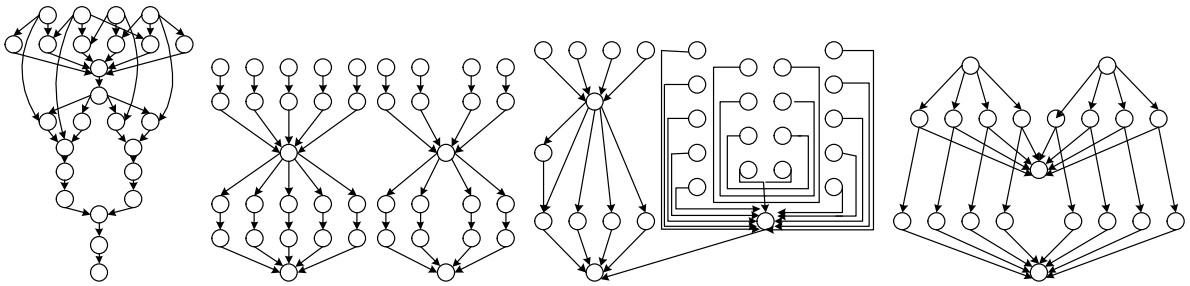
Then, we transfer the optimization problem as follows.

$$\text{Minimize: } cost(t_i) \quad (18)$$

$$\text{Subject to: } T_{rent}(t_i, VM(k)) \leq T_{subm}(t_i) \quad (19)$$

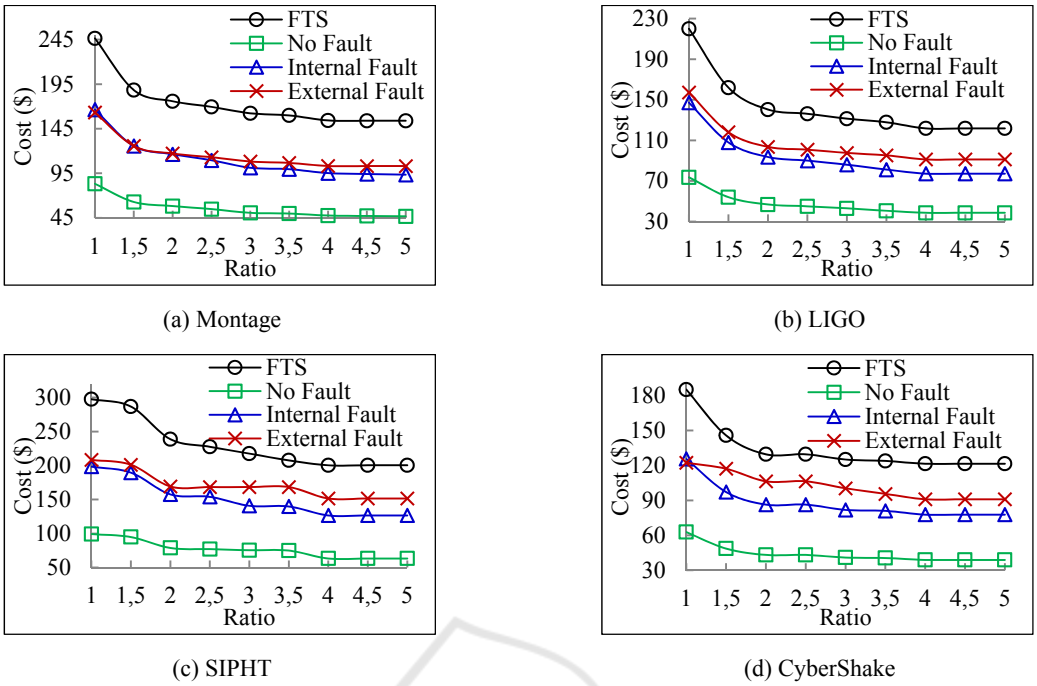
$$\text{Eq. (16)} \quad (20)$$

The new problem means that as for each task, we find a $VM(k)$ for task t_i to satisfy Eqs. (19) and (20), and minimize the $cost(t_i)$. Eqs. (19) and (20) represent the sub-makespan constraint and the conditions for successfully executing respectively. First, we find a VM set $VM_{feasible}(t_i)$ of task t_i which can satisfy the Eq. (19). Then, from the set $VM_{feasible}(t_i)$, the optimal $VM_{opt}(k)$ is selected that the Eq. (20) is met and the cost of task t_i is minimized. Here, the enumeration is used to find the optimal VM type effectively. The time complexity of enumeration depends on the number of VMs. Overall, the pseudo code of our proposed FTS algorithm is described in Fig. 1. We can see that the time complexity of computing minimum execution time is $O(n)$ (lines 1-3). Then, the worst time complexity of calculating the optimal VM type is $O(n^2)$ (lines 5-11). As a result, the time complexity of FTS algorithm is $O(n^2)$.



(a) Montage (b) LIGO (c) SIPHT (d) CyberShake

Figure 2: Structures of real-world scientific workflows.



(a) Montage

(b) LIGO

(c) SIPHT

(d) CyberShake

Figure 3: Workflow cost with different deadlines.

5 SIMULATION EXPERIMENTS

In this section, we simulate a series of experiments to evaluate the performance of our FTS approach. First, we introduce the experiment setup. Then, the simulation results on the basis of four real-world workflows are discussed.

5.1 Experiment Setup

The simulation is run on a machine with an Intel Core i7 4 cores and 4 GB of RAM. All algorithms are implemented in Python 3.5. Four scientific workflow models from different areas, such as Montage, LIGO, SIPHT and CyberShake, are used in our experiments are shown in Fig. 2 (Zhu et al., 2016; Li et al., 2016, 2017). For each task in a workflow, the size of input/output data and the workload are all according to the uniform distribution in the range [10, 100] GB and [1, 64] CU respectively. The bandwidth between two VMs is 0.1GB/s. Let $ratio = deadline/makespan^{\min}$ is the rate of predefined deadline to minimum makespan. In order to set the proper deadline for each workflow, we must have $ratio \geq 1$.

We also consider other three workflow scheduling scenario to compare with FTS approach. The comparison algorithms are presented as follows.

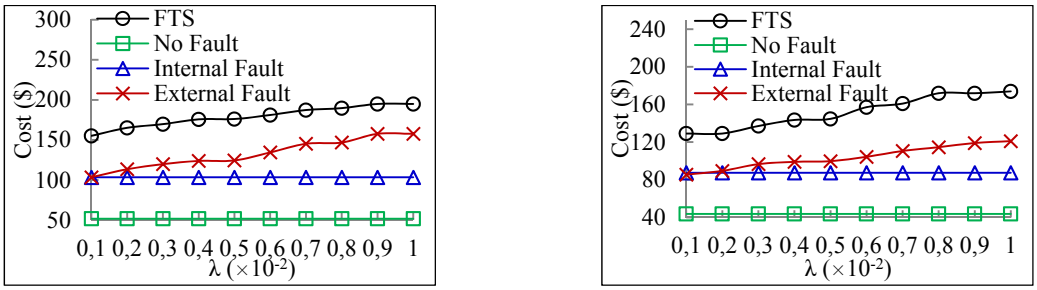
No Fault: That means no internal and external failures exist in the cloud computing environment, and users need not to employ any fault-tolerant strategies.

Internal Fault: In this case, only host fault will happen during the tasks executing.

External Fault: Users only consider the malicious attack failure in this assumption. Then, the security time overhead should be integrated into the makespan and cost computation.

5.2 Simulation Results

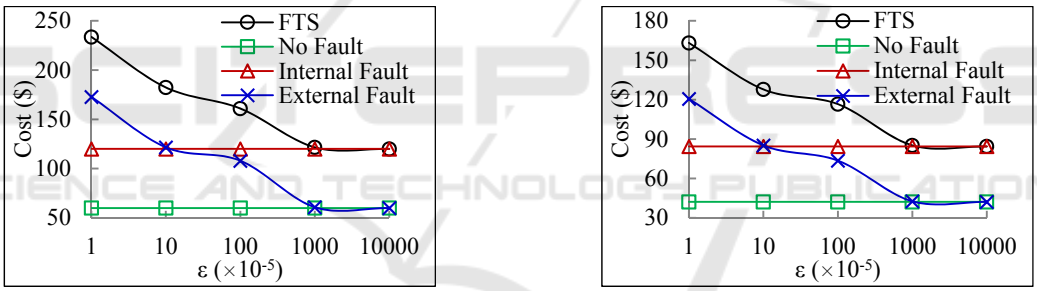
The experiment results of four scientific workflows on cost under different deadlines are shown in Fig. 3, where the ratio is from 1 to 5 with the increment of 0.5, $\lambda = 0.005$ and $\varepsilon = 10^{-4}$. We can see from the four figures that the workflow cost become lower as the ratio increases. This is because that a larger workflow deadline will generate larger sub-makespan for tasks, then a task will map to the low performance VM with low cost. Moreover, No Fault algorithm has the lowest cost. The cost of Internal Fault algorithm is double of the No Fault algorithm, for it always have two copies tasks to execute. The cost of External Fault algorithm is roughly same to Internal Fault algorithm. Our proposed FTS algorithm performs worst in scheduling cost. This is due to the fact that internal and external faults are



(a) Montage (b) LIGO

(c) SIPHT (d) CyberShake

Figure 4: Workflow cost under different λ.



(a) Montage (b) LIGO

(c) SIPHT (d) CyberShake

Figure 5: Workflow cost under various ε.

considered at the same time in FTS algorithm. In order to meet the reliability execution of workflow, multiple copies of a task, up to the failure coefficient, are needed to execute. Although FTS algorithm has the most workflow cost, only it can ensure the successful execution of workflow even in the presence of host fault and malicious attack simultaneously in the cloud computing environment.

We plot the workflow cost under different λ for FTS algorithm and the peer algorithms are presented

in Fig. 4, where $ratio = 2.5$ and $\varepsilon = 10^{-4}$. As we can see from the figures, No Fault algorithm and Internal Fault algorithm are independent of λ. With the larger λ, the cost of FTS algorithm and External Fault algorithm increase with the same pace. This results from the fact that a larger value of λ means the cloud computing platform will suffer from more external malicious attacks. Then, a user should schedule more copies of tasks to VMs that induce more cost. Similarly, the FTS algorithm performs the worst on execution cost, the reason of which is that FTS can tackle the internal and external faults more efficiently than any other comparison algorithms.

Fig. 5 illustrates the workflow cost of four algorithms under different self-defined ε, where $ratio = 2.5$ and $\lambda = 0.005$. From the four workflow structures, it can be seen that the curves of No Fault algorithm and Internal Fault algorithm are flat. The rationale is that No Fault algorithm and Internal Fault algorithm are independent with the parameter ε. The cost curves of FTS algorithm and External Fault algorithm are decreasing when parameter ε become large. The reason is that less task copies will demand when parameter ε is large according to Eq. (16), which introduces less VM cost. Moreover, the cost of FTS algorithm and External Fault algorithm perform equal with the cost of Internal Fault algorithm and No Fault algorithm respectively when $\varepsilon = 0.01$ and $\varepsilon = 0.1$. As for External Fault algorithm and No Fault algorithm, when the ε is large enough, the failure probability produced by the current failure coefficient λ is already less than ε, hence no additional copy is needed. Thus, External Fault algorithm and No Fault algorithm have the same workflow cost. Also, the same explanation can be used for FTS algorithm and internal algorithm.

6 CONCLUSION AND FUTURE WORK

This paper we proposed a fault-tolerant scheduling for scientific workflow in cloud computing environment. The purpose of FTS algorithm is to minimize the workflow cost with the deadline constraint, which is based on tasks replication method (one of the widely used fault tolerant mechanisms). As far as we known, it is the first time that both failure models (i.e. internal fault and external fault) are considered in cloud workflow scheduling problem. The simulation results with real-world scientific workflow models show that only FTS algorithm can ensure the successful execution of workflow, although it has the most workflow cost.

For the future work, we will extend our fault-tolerant workflow scheduling strategy considering unstable computing environments. Another direction of our future work is to design an energy efficient fault-tolerant task scheduling algorithm from the perspective of cloud providers.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation of China (No. 61572162, 61572251, 61702144), the Zhejiang Provincial National Science Foundation of China (No. LQ17F020003), the Zhejiang Provincial Key Science and Technology Project Foundation (NO.2018C01012), the National Key R&D Program of China (2016YFC0800803), the Fundamental Research Funds for the Central Universities. Hua Hu is the corresponding author.

REFERENCES

Amazon EC2, 2017. http://aws.amazon.com/ec2/.

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.

Chen, W., da Silva, R. F., Deelman, E., Fahringer, T., 2016. Dynamic and Fault-Tolerant Clustering for Scientific Workflows. *IEEE Transactions on Cloud Computing*, 4 (1), 49-62.

Dean, J., 2009. Designs, Lessons and Advice from Building Large Distributed Systems. LADIS, 2009, http://www.cs.cornell.edu/projects/ladis2009/program.htm#keynote3.

Durillo, J. J., Prodan, R., Fard, H. M., 2012. Moheft: A Multi-Objective List-Based Method for Workflow Scheduling. *International Conference on Cloud Computing Technology and Science (CloudCom 2012)*, pp. 185-192.

Fard, H. M., Prodan, R., Barrionuevo, J. J. D., Fahringer, T., 2012. A Multi-objective Approach for Workflow Scheduling in Heterogeneous Environments. *IEEE/ACM International Symposium on Cluster (CCGRID 2012)*, pp. 300-309.

Fard, H. M., Prodan, R., Fahringer, T., 2013. A Truthful Dynamic Workflow Scheduling Mechanism for Commercial Multicloud Environments. *IEEE Transactions on Parallel and Distributed Systems*, 24 (6), 1203-1212.

Foster, I., Zhao, Y., Raicu, I., Lu, S.Y., 2008. Cloud Computing and Grid Computing 360-Degree Compared. *IEEE Grid Computing Environments (GCE08) 2008 and IEEE/ACM Supercomputing 2008*.

Ghosh, S., Melhem, R., Mosse, D., 1997. Fault-Tolerance Through Scheduling of Aperiodic Tasks in Hard Real-Time Multiprocessor Systems. *IEEE Transactions on Parallel and Distributed Systems*, 8 (3), 272-284.

Jeannot, E., Saule, E., Trystram, D., 2012. Optimizing Performance and Reliability on Heterogeneous Parallel Systems Approximation Algorithms and Heuristics. *Journal of Parallel and Distributed Computing*, 72 (2), 268-280.

Kashlev, A., Lu, S.Y., 2014. A System Architecture for Running Big Data Workflows in the Cloud. *2014 IEEE International Conference on Services Computing (SCC)*, pp. 51-58.

Kyriazis, D., Tserpes, K., Menychtas, A., Litke, A., Varvarigou, T., 2008. An Innovative Workflow Mapping Mechanism for Grids in the Frame of Quality of Service. *Future Generation Computer Systems*, 24 (6), 498-511.

Li, Z., Ge, J., Hu, H. Y., Song, W., Hu, H., Luo, B., 2017. Cost and Energy Aware Scheduling Algorithm for Scientific Workflows with Deadline Constraint in Clouds. *IEEE Transactions on Services Computing*.

Li, Z., Ge, J., Yang, H., Huang, L., Hu, H. Y., Hu, H., Luo, B., A Security and Cost Aware Scheduling Algorithm for Heterogeneous Tasks of Scientific Workflow in Clouds. *Future Generation Computer Systems*, 65, 140-152.

Manimaran, G., Murthy, C. S. R., 1998. A Fault-tolerant Dynamic Scheduling Algorithm for Multiprocessor Real-time Systems and Its Analysis. *IEEE Transactions on Parallel and Distributed Systems*, 9 (11), 1137-1152.

Mao, M., Humphrey, M., 2012. A Performance Study on the VM Startup Time in the Cloud. *IEEE 5th International Conference on Cloud Computing*, pp. 423-430.

Mell, P., Grance, T., 2009. The Nist Definition of Cloud Computing. *National Institute of Standards and Technology*, 53 (6), p. 50.

Plankensteiner, K., Prodan, R., 2012. Meeting Soft Deadlines in Scientific Workflows Using Resubmission Impact. *IEEE Transactions on Parallel and Distributed Systems*, 23 (5), 890-901.

Qiu, X., Dai, Y., Xiang, Y., Xing, L., 2017. Correlation Modeling and Resource Optimization for Cloud Service With Fault Recovery. *IEEE Transactions on Cloud Computing*.

Rodriguez, M. A., Buyya, R., 2014. Deadline based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds. *IEEE Transactions on Cloud Computing*, 2 (2), 222-235.

Sun, G., Chang, V., Yang, G., Liao, D., 2017. The Cost-efficient Deployment of Replica Servers in Virtual Content Distribution Networks for Data Fusion. *Information Science*.

Sun, G., Liao, D., Zhao, D., Xu, Z., Yu, H., 2016. Live Migration for Multiple Correlated Virtual Machines in Cloud-based Data Centers. *IEEE Transactions on Services Computing*.

Vinay, K., Dilip Kumar, S. M., 2017. Fault-tolerant Scheduling for Scientific Workflows in Cloud Environments. *IEEE 7th International Advance Computing Conference (IACC)*, pp. 150-155.

Wang, J., Bao, W., Zhu, X., Yang, L. T., Xiang, Y., 2015. FESTAL: Fault-Tolerant Elastic Scheduling Algorithm for Real-Time Tasks in Virtualized Clouds. *IEEE Transactions on Computers, 64 (9)*, 2545-2558.

Xie, T., Qin, X., 2006. Scheduling Security-critical Real-Time Applications on Clusters. *IEEE Transactions on Computers*, 55 (7), 864-879.

Xie, T., Qin, X., 2008. Security-aware Resource Allocation for Real-Time Parallel Jobs on Homogeneous and Heterogeneous Clusters. *IEEE Transactions on Parallel and Distributed Systems*, 19 (5), 682-697.

Yao, G., Ding, Y., Hao, K., 2017. Using Imbalance Characteristic for Fault-Tolerant Workflow Scheduling in Cloud Systems. *Transactions on Parallel and Distributed Systems*.

Zeng, L. F., Veeravalli, B., Li, X. R., 2015. SABA: A Security-Aware and Budget-aware Workflow Scheduling Strategy in Clouds. *Journal of Parallel and Distributed Computing*, 75, 141-151.

Zhao, Y., Fei, X., Raicu, I., Lu, S., 2011. Opportunities and Challenges in Running Scientific Workflows on the Cloud. *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 455-462.

Zhao, Y., Li, Y., Tian, W., Xue, R., 2012. Scientific-Workflow-Management-as-a-Service in the Cloud. *2012 Second International Conference on Cloud and Green Computing*, pp. 97-104.

Zhu, X., Qin, X., Qiu, M., 2011. QoS-Aware Fault-Tolerant Scheduling for Real-Time Tasks on Heterogeneous Clusters. *IEEE Transactions on Computers*, 60 (6), 800-812.

Zhu, X., Wang, J., Guo, H., Zhu, D., Yang, L. T., Liu, L., 2016. Fault-Tolerant Scheduling for Real-Time Scientific Workflows with Elastic Resource Provisioning in Virtualized Clouds. *IEEE Transactions on Parallel and Distributed Systems*, 27 (12), 3501-3517.

Zhu, Z., Zhang, G., Li, M., Liu, X., 2016. Evolutionary Multi-Objective Workflow Scheduling in Cloud. *IEEE Transactions on Parallel and Distributed Systems*, 27 (5), 1344-1357.