Information Sciences 393 (2017) 47–65

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fault-tolerant elastic scheduling algorithm for workflow in Cloud systems

Yongsheng Ding[a,*], Guangshun Yao[a,b,*], Kuangrong Hao[a]

[a] *Engineering Research Center of Digitized Textile and Apparel Technology, Ministry of Education, College of Information Science and Technology, Donghua University, Shanghai 201620, China*
[b] *College of Computer and Information Engineering, Chuzhou University, Chuzhou, Anhui 239000, China*

ARTICLE INFO

Article history:
Received 13 January 2016
Revised 23 January 2017
Accepted 29 January 2017
Available online 31 January 2017

Keywords:
Fault-tolerant
Workflow
CLOUD systems
Elastic scheduling
Primary and backup scheduling
Virtualization

ABSTRACT

With the increasing functionality and complexity of Cloud systems, fault tolerance becomes an essential requirement for tasks executed in Cloud systems, especially for the workflow with dependent relationship among tasks. The existing fault-tolerant scheduling algorithms cannot be used for workflow in Cloud systems directly, which have two key features: virtualization and elasticity. In this paper, we propose an offline fault-tolerant elastic scheduling algorithm for workflow in Cloud systems (FTESW). After analyzing the constrains of primary-backup scheduling in Cloud systems caused by the dependence among tasks in the submitted workflow, an elastic mechanism in the context of fault tolerance is designed to dynamically adjust the resource provisioning based on the resource request by adopting the technology of resource migration. Then, the FTESW is proposed to achieve both fault tolerance and high resource utilization for workflow in Cloud systems. To verify the effectiveness of the proposed FTESW, a series of simulation experiments are conducted on both randomly generated workflows and real-world workflows. The simulation results demonstrate that the proposed FTESW is able to effectively provide a corresponding fault-tolerant scheduling strategy for workflow with high resource utilization in Cloud systems and has better performance than the corresponding competitors.

1. Introduction

Cloud computing, usually supported by virtualized data centers, has emerged as an attractive platform for increasingly diverse applications. It allows for low-entry costs, reduced cost of maintaining IT infrastructure, and on demand heterogeneous services and resources provisioning in a pay-as-you-go model [8,13,14]. Empowered by the flexibility in obtaining and releasing computing resources in a cost-effective manner and virtualization technology, Cloud systems give customers the illusion of unlimited computing resources and have been widely adopted by increasing customers for their tasks. While the benefits are immense, the Cloud systems have a high resources failure probability due to the increased functionality and complexity [18,25,26,32]. Such faults can result in frequent performance degradation, premature termination of execution, data corruption and loss, and cause a devastating influence to the executing of submitted tasks, especially for the workflow with dependent relationship among tasks since a fault delay in finishing one task caused by can propagate widely, causing

[*] Corresponding authors.
E-mail addresses: ysding@dhu.edu.cn (Y. Ding), yaoguangs@126.com (G. Yao).

http://dx.doi.org/10.1016/j.ins.2017.01.035

delays among many tasks on the other related resources. Therefore, how to provide a fault-tolerant mechanism is mandatory for tasks executed in Cloud systems.

Fault-tolerant scheduling is an effective approach to achieve fault tolerance by allocating multiple copies of each task to different computing nodes [15,16,28]. Among the fault-tolerant scheduling approaches, the primary-backup (PB) model is a popular scheme and has drawn a great deal of attentions [11,19,30]. In the PB model, two copies of each task, namely, the primary one and the backup one, are allocated to two different computing instances for fault tolerance. If the node with the primary copy encounters a fault, the backup copy can still be executed on a different node to guarantee the successful completion of the task.

During the last decades, lots of studies have been conducted to design effective fault-tolerant scheduling approaches based on the PB model in distributed systems. Ghosh et al. proposed two techniques, deallocation and overloading, to improve schedulability while providing fault-tolerance with low overhead [15]. Manimaran and Murthy extended the strategy in [15] by taking the resource constraints among tasks into consideration, and partitioned processors into multiple groups to tolerate more than one fault at a time [24]. Al-Omari et al. designed a PB overloading technique that allowed the primary copy of one task to overlap with the backup copy of another task for the purpose of high schedulability [3]. Zhu et al. proposed a QoS-aware fault-tolerant scheduling algorithm and an adaptive service-aware fault-tolerant scheduling algorithm based on overlapping technology for independent tasks within heterogeneous clusters [37,38]. By juxtaposing the strong points of neighbor-based and cluster-based load-balancing methods in Grid systems, Balasangameshwara and Raju developed a load-balancing approach and integrated it with the fault-tolerant scheduling namely MinRC, as such proposed a performance-driven fault-tolerant load-balancing algorithm for independent tasks [6]. Luo et al. proposed two novel fault-tolerant techniques, which were integrated with fixed-priority-based scheduling algorithms, for periodic real-time tasks [23]. In order to save the energy consumed by the Cloud systems, Cui et al. designed an energy-aware and fault-tolerant scheduling algorithm by using the dynamic voltage and frequency scaling (DVFS) technology to adjust the speed of computing node with backup copy [11]. Although the above existing approaches have improved the performance and schedulability of systems greatly from different perspectives, they are only used for independent tasks and cannot be used for workflow applications where tasks have precedence constraints.

As for the fault-tolerant scheduling for workflow applications, Qin and Jiang proposed a new overlapping scheme, which allowed the backup copy of a task to overlap with the primary copies of its successors [27]. After identifying two crucial limitations of scheduling primaries and backups of dependent tasks in Grid systems, Zheng et al. proposed two fault-tolerant scheduling algorithms based on the PB model [36], namely MRC-ECT and MCT-LRC, to schedule backups of independent and dependent jobs, respectively. For the Grid computing systems with dedicated communication devices, Zheng and Veeravalli designed two communication-aware fault-tolerant scheduling algorithms based on the PB model to avoid the influence caused by the processors faults and communication delays [35]. Jing and Liu designed a fault-tolerant scheduling algorithm called CCRH based on the PB model for workflow in Cloud systems [19]. Although the above algorithms have exhibited good performances for fault-tolerant scheduling for workflow from different aspects, however, when taking the features of Cloud systems into account: 1) The virtualization technology, which makes virtual machine (VM) become the basic computing instance rather than host and enables VMs to migrate across the multiple hosts, is not considered, and the constraints of VM migration are not analyzed when using the PB model; 2) The elasticity of the systems, a new resource provisioning strategy in Cloud systems, has not either been studied in the above literatures.

Recently, Wang et al. proposed an elastic resource provisioning mechanism in the context of fault tolerance to improve the resource utilization by incorporating the above features of Cloud systems [30]. However, it is only used for independent tasks. Bala and Chana used VMs migration that migrated the faulty VMs to another host to achieve fault tolerance for workflow tasks [5]. Nonetheless, it is only used for the fault caused by overutilization of resources and not suitable for the host fault, which makes the migration of VMs infeasible. Moreover, the elasticity of resource provisioning is not considered in this approach.

In this paper, we design an offline fault-tolerant elastic scheduling algorithm for workflow in Cloud systems (FTESW) based on the PB model by taking both virtualization and elasticity into consideration. Firstly, we analyze the limitations of PB scheduling and resource migration in Cloud systems for the precedence constrained tasks in workflow. Then, we propose an elastic resource provisioning mechanism to optimize resource utilization while supporting fault tolerance in Cloud systems. Based on the above work, the FTESW, including both primary scheduling strategy and backup scheduling strategy, is proposed. In order to verify the effectiveness of the proposed FTESW, we compare it with some other fault-tolerant scheduling algorithms for workflow in Cloud systems. Simulation results highlight the performance of the proposed approach.

The main contributions of this paper are as follows. Firstly, the constrains that must be satisfied for the fault-tolerant scheduling based on PB model are analyzed to determine the suitable located VM and start time of dependent tasks in Cloud systems. Secondly, the rules about resource migration for dependent tasks in workflow are dissected to improve the resource utilization of Cloud systems in the context of fault tolerance. Thirdly, an elastic resource provisioning mechanism is designed to adjust the active resources dynamically according to the scheduling request by adopting the technology of resource migration in Cloud systems. Finally, the FTESW, including both elastic primary scheduling strategy and elastic backup scheduling strategy, is proposed for fault-tolerant workflow scheduling in Cloud systems with high resource utilization.

The rest of this paper is organized as follows. In Section II, we depict the workflow, Cloud and fault model used in this paper. Then, the constrains about the PB scheduling and resource migration for the precedence constrained tasks are