Knowledge-Based Systems 99 (2016) 39–50

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

An immune system-inspired rescheduling algorithm for workflow in Cloud systems

Guangshun Yao[a,b], Yongsheng Ding[a,*], Lihong Ren[a], Kuangrong Hao[a], Lei Chen[a]

[a] Engineering Research Center of Digitized Textile and Apparel Technology, Ministry of Education; College of Information Science and Technology, Donghua University, Shanghai 201620, P R China

[b] College of Computer and Information Engineering, Chuzhou University, Chuzhou 239000, Anhui, P R China

ARTICLE INFO

Article history:
Received 28 June 2015
Revised 27 January 2016
Accepted 29 January 2016
Available online 18 February 2016

Keywords:
Cloud computing systems
Resource rescheduling
Workflow
Immune systems
Fault tolerant

ABSTRACT

Due to the increasing functionality and complexity of Cloud computing systems, the resource failures, including unpredictable crash and performance degradation about resource availability, are inevitable. So a failure-aware resource provisioning algorithm that is capable of offering corresponding strategies immediately after failures happened is paramount. In this paper, an immunological mechanism inspired rescheduling algorithm is proposed for workflow in Cloud systems (IRW). There are four units to imitate the immune system in the IRW algorithm. The surveillance unit monitors possible faults for each Virtual Machine (VM) in resources pool. Once a resource fault is detected, the response unit is triggered to search an appropriate strategy either in the memory unit or in the learning unit for rescheduling the available resources. The available resources are clustered into multiple clusters to narrow the search scope in the learning unit. If none of available VMs can meet the Quality of Services, a new VM is created for the faulty resource. To verify the effectiveness of the proposed IRW, a series of simulation experiments are conducted on both real world workflows with different structures and randomly generated workflows. The results demonstrate that the IRW is able to effectively provide corresponding rescheduling strategies for resource failures and the experiments also highlight the better performance of the proposed approach than that of corresponding algorithms under different situations.

I. Introduction

Cloud computing promises to provide large-scale heterogeneous computational resources to customers over network in a pay-as-you-go model [1]. The flexibility in obtaining and releasing computational resources in a cost-effective manner has resulted in a wide adoption of the Cloud computing paradigm. Infrastructure as a Service (IaaS) is one of the most common Cloud service models, which provides customers with the abilities to provide or release pre-configured computational resources from a Cloud infrastructure. In this model, customers can access to almost unlimited number of resources for their tasks. With the increasing submitted tasks, how to schedule the available computational resources for these tasks becomes a critical part of Cloud systems as it has significant influence on the efficiency of Cloud computing. Meanwhile, due to the increased functionality and complexity of the Cloud systems, resource faults are inevitable during the process of services [2]. Such faults can result in frequent performance degradation, premature termination of execution, data corruption and loss, and cause a devastating loss of customers and revenue. Therefore, providing a fault-tolerant mechanism is necessary for efficiently utilizing the computational resources in Cloud systems.

In the context of workflow task, it is even more difficult because a workflow task usually contains a great number of jobs that have precedence constraints, where the input of some jobs may depend on the output of the others. In recent decades, lots of research works have been conducted about workflow scheduling in Cloud systems [3-5]. However, the fault-tolerant mechanism has rarely been simultaneously considered in these works.

The two fundamental and widely recognized techniques to support the fault in distributed environments are replication and rescheduling. And they are complementary and can be used in cooperation: the replication is a method suited to the tasks scheduling phase, while the rescheduling is the one mostly applicable during execution [6]. In this paper, we focus on the rescheduling method.

Rescheduling is an effective method to tolerate the fault of computational resources [7], especially for the tasks with enough laxity. It mainly refers to the scheme of reallocating computational resources for some tasks when there is any malfunction of the

* Corresponding author. Tel.: +86 21 67792301.
E-mail address: ysding@dhu.edu.cn (Y. Ding).

http://dx.doi.org/10.1016/j.knosys.2016.01.037
0950-7051/

systems. Unlike the replication scheduling method mentioned above, the resource rescheduling method can reduce the resources consumption distinctly. It has been considered by a number of projects for Grid Computing, such as AppLeS [8], Condor-G [9], and Nimrod-G [10].

Recently, some methods about rescheduling work [7, 11-13] have been proposed. Plankensteiner et al. [7] proposed a dynamic enactment and rescheduling heuristic able to execute workflows with fault tolerance and taken into account soft deadlines simultaneously, while it needed some resources for jobs replication to meet the deadline constraints. Cao et al. [11] designed three schemes for Virtual Machines (VMs) rescheduling in Cloud computing systems. After detecting a VM was crashed, these schemes firstly stored the executing jobs on this VM and waited for a certain time to check whether this VM could be repaired. If this VM could be repaired, the stored jobs were executed on this VM. Otherwise, the stored jobs were migrated to another VM. Chen et al. [12] proposed an efficient rescheduling heuristic with the support of reservation adjustment for multiple workflows submitted at different time, while it supposed all resources with uncertain performance were available all the time and did not consider the crashed resource. Olteanu et al. [13] designed four generic rescheduling strategies by combining a wide variety of scheduling heuristics together for multiple workflows scheduling, such as retry, alternate resource, rescue file and user-defined exception handling. However, the processes of both retry and rescue file needed lots of time and the strategy about user-defined exception handling was unpractical in real world. Moreover, the system administrator had to pre-configure the resources for rescheduling based on factors such as the number and characteristics of available resources, and the structure and attributes of the graph task that need to be scheduled. Sakellariou et al. [14] considered rescheduling at some carefully selected points along execution. After the initial scheduling was obtained, a set of unfinished tasks were selected for rescheduling if the run time performance variation exceeded a predefined threshold. All the above rescheduling algorithms could improve the performance of scheduling and the utilization efficiency of computational resources. However, they did the identical work when the same failure happened again and their efficiency could be improved greatly.

Besides the replication and rescheduling based fault-tolerant scheduling, many other algorithms have also being proposed to deal with fault-tolerant scheduling in distributed systems. Some researchers [15-17] started to investigate how to manage and predict failures in complex infrastructures for the fault. However, building the model used in these approaches can be a very difficult task that often required the traces of failure data about the specific target environment. On the other hand, commercial Cloud providers, such as Amazon, do not disclose any information regarding their infrastructure, and failure traces are often a closely guarded secret. Jhawar et al. [18] advocated a new dimension where the tasks deployed in a Cloud computing infrastructure could obtain the required fault tolerance properties from a third party. Nevertheless, selecting of the corresponding third party is difficult for normal users and the strategies adopted in the third party are also based on replication. Zheng [19] and Qiu [20] proposed a ranking-based method in which all components in Cloud systems are ranked according to their invocation structures and invocation frequencies. Based on the ranking results, an optimal algorithm was derived to determine the fault tolerance strategies for different components. However, the precise ranking is very hard to achieve and requires not only a deep knowledge of the behavior about the target infrastructure, but also years of tracing data of the specific system.

In this paper, an immune system inspired failure-aware rescheduling algorithm for workflow task in Cloud systems (IRW) is designed. In the last decade, the artificial immune system has drawn significant attention [21-24] and been used in many domains successfully, such as fault detection and diagnosis [25, 26], intrusion detection [27, 28], cigarette production scheduling [29], web service composition and management [30], automatic diagnosis of electrocardiogram arrhythmia [31], disturbances control in public transportation systems [32], and industrial control [33, 34]. The immune system has many functions, including the immune surveillance, the immune response, the immune memory and the learning ability, and works cooperatively to protect the biosystem in health. Through analyzing, it can be found that the immune system has many features similar to the objectives of the resource rescheduling strategy. In the IRW, we design a surveillance unit to imitate the immune surveillance process, a response unit to imitate the immune response process, a memory unit to imitate the immune memory mechanism, and a learning unit to imitate the learning ability of the immune system. For decreasing the extra overhead caused by rescheduling, we classify the available resource into multiple clusters to narrow search scope in the learning unit and take the communication and execution time of job caused by rescheduling as the affinity to select suitable candidate computational resource for the fault in the response unit. If the candidate cannot meet the Quality of Service (QoS), a new VM is created on one of the active hosts or a turned on host. By treating the rescheduling process as a biological immune system, all these units collaborate with each other and realize the rescheduling process. In order to verify the effectiveness of the proposed IRW, we do some simulations and compare it with some rescheduling algorithms. Simulation results highlight the performance of the proposed approach.

The main contributions of this paper are as follows. A biological immune system inspired rescheduling algorithm, called IRW, is proposed for workflow in Cloud systems. The IRW can achieve better reschedule performance under the conditions of resources failures, not only crashed but also performance degraded resources in Cloud systems. By treating the rescheduling process as an immune system, the proposed IRW can provide better rescheduling strategies for unknown fault cases and more quickly response to the same failures than the existing rescheduling algorithms. In order to offer information of available resources for rescheduling, a manager server is designed to detect the change of resources through collaboration of the distributed surveillance units. Besides rescheduling the existing available resources, creating new VMs is also considered in IRW if necessary.

The rest of this paper is organized as follows. In Section 2, we depict the workflow and Cloud data center model used in this paper. After introducing the biological immune system, the immune system inspired rescheduling algorithm for workflow in Cloud systems is proposed in Section 3. The evaluation of our algorithm and the analysis of the obtained results are included in Section 4. Finally, the main conclusions and future work are presented in Section 5.

2. Workflow and Cloud systems model

2.1. Task model for workflow

In this paper, we use the Directed Acyclic Graph (DAG) to represent the workflow tasks submitted to Cloud systems. A DAG $V = (T, E)$ consists of R jobs $T = \{t_1, t_2, \ldots, t_R\}$ which are interconnected through control flow and data flow such as

$$E = \{(t_i, t_j, Data_{ij}) \big| (t_i, t_j) \in T \times T, i \neq j\},$$

where $Data_{ij}$ represents the size of which needs to be transferred from job t_i to t_j. We also use $pred(t_i)$ and $succ(t_i)$ to denote the set of predecessors and successors of job t_i, which means $pred(t_i)$ must