Service Cost Effective and Reliability Aware Job Scheduling Algorithm on Cloud Computing Systems

Xiaoyong Tang, Yi Liu, Zeng Zeng, *Senior Member, IEEE*, and Bharadwaj Veeravalli, *Senior Member, IEEE*

Abstract—Nowadays, increasing number of services are provided to individuals and organizations through cloud computing systems in a *pay-as-you-use* model. This business service paradigm encounters several cloud Quality of Service (QoS) challenges, such as reliability, cost, and response time. The most common mechanism to improve cloud service reliability is a primary/backup (PB) fault-tolerant technique. However, this reliability enhancement technique inevitably results in multiple replications, which lead to high service cost. In recognition of these challenges, we first build a cloud computing systems resources management architecture. Then, we analyze the cloud service execution reliability on the physical resources of a VM and used a CUDA (Compute Unified Device Architecture)-enabled parallel two-dimensional long short-term memory neural network to predict the software faults of a cloud VM. Third, we propose an effective primary/backup cloud service cost calculation approach. To overcome the cloud service response time constraint, we integrate a response time slack factor into this method. Fourth, we formulate the cloud service reliability and cost aware job scheduling problem, which aims at minimizing the total cloud service cost and rejection rate, and improving the system reliability. Fifthly, a heuristic greedy reliability and cost aware job scheduling (RCJS) algorithm is proposed. Finally, a performance evaluation is conducted and the experimental results demonstrate that our proposed RCJS algorithm significantly outperforms optimal redundant VM placement (OPVMP), MIN-MIN algorithms in terms of average service cost and rejection rate. This algorithm also demonstrates good trade-off of reliability when compared to the other two algorithms and is suitable for cloud services with high reliability and low-cost requirements.

Index Terms—Cloud computing systems, service, reliability, cost, job scheduling

1 Introduction

With the rapid deployment of on-demand virtualized IT infrastructures, such as servers, storage, and databases, an increasing number of cloud services are provided to individuals and organizations through wired and wireless Internet in a *pay-as-you-use* model [1]. This is owing to cloud computing, the most prominent resource delivery model, in which all resources, such as machines, networks, applications, software, platforms, can be delivered as services [2]. Similar to social economic services, cloud Quality of Service (QoS) requirements, such as security, reliability, energy consumption, and cost, are the key challenges for cloud service providers, which have attracted increasing attention from businesses and academia [3], [4]. Among all these challenges, reliability and cost are two crucial concerns for both cloud users and providers [5].

There are several types of accidents that can affect cloud service reliability including service delay, service interruption, service error, data loss, and long response time [6]. These events not only influence the service reliability of cloud computing suppliers, but can also result in significant economic damages and certain harmful social impacts, such as the recent outages in Netflix, AWS, Huawei, Twitter, Tencent, and Facebook [7]. Ponemon institute reported that the average cost of the IT computing system outages was approximately 740,357 in 2016 [8]. This is primarily because the cloud computing systems are confronted by inevitable risks and failures, such as software faults, hardware faults, virtual machine (VM) failures. These failures may in turn delay or result in cloud Service Level Agreement (SLA) violations and damage the entire cloud service processing system. Moreover, certain cloud services consist of tens of thousands of fine granularity tasks and the failure of any of these jobs can affect the reliability of the entire cloud service [9].

One of renowned techniques to address cloud service reliability issues is the fault-tolerant technique, which ensures cloud service continuity by utilizing one or more backups when a failure occurs [5], [10]. The cloud system fault tolerance can be achieved by scheduling services (or jobs) to

• *Xiaoyong Tang and Yi Liu are with the School of Computer and Communications Engineering, Changsha University of Science and Technology, Changsha 410114, China. E-mail: {tang_313, yiliu_852}@163.com.*
• *Zeng Zeng is with Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore 138632. E-mail: zengz@i2r.a-star.edu.sg.*
• *Bharadwaj Veeravalli is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576. E-mail: elebv@nus.edu.sg.*

Manuscript received 14 July 2021; revised 25 October 2021; accepted 17 December 2021. Date of publication 23 December 2021; date of current version 7 June 2023.
This work was supported in part by the National Natural Science Foundation of China under Grant 61972146, in part by Hunan Provincial Key Research and Development Program under Grant 2018GK2055, and in part by the Hunan Provincial Natural Science Foundation of China under Grant 2020JJ4376.
(Corresponding author: Zeng Zeng.)
Recommended for acceptance by S. Khan.
Digital Object Identifier no. 10.1109/TCC.2021.3137323

different VMs. The cloud service may have multiple replications; consequently, it is tolerant to a few failures. The conventional fault-tolerance mechanism is a primary/backup (PB) approach, where each cloud service is scheduled on a primary VM and a backup VM [11]. However, these multiple replications-based reliability enhancement technology requires the utilization of more redundant system VMs, which inevitably results in high cloud service cost.

An effective solution to reduce cost is to only backup partial services in the primary/backup (PB) approach. In practical terms, the reliability of the modern cloud computing systems is relatively high, and most requests of cloud services requests can be successfully completed in general. Therefore, we only need to analyze and predict low-reliability cloud virtual machine services processed by the cloud VM, and backup these services to other VMs. In such case, we can ensure the reliability of cloud services and save considerably reduce the corresponding services costs as much as possible. The key aspect of this problem is to accurately analyze the reliability of the physical resources of the VMs in cloud computing systems VM physical resources and predict the VM software fault of the VMs.

To overcome the aforementioned challenges, this paper presents a cloud resources management architecture, analyzes cloud service execution reliability and cost, and proposes a heuristic greedy reliability and cost aware job scheduling (RCJS) algorithm. In this paper, we present multifold fundamental contributions, which can be summarized as follows:

- First, we build a cloud computing systems resources management architecture, which consists of cloud users, service requests queue, service reliability analysis, resource management and scheduling model, VMs layer, and physical resources layer.
- Second, we analyze the cloud service execution reliability caused by the physical and software resources of VM, and propsoe a CUDA-enabled parallel two-dimensional long-short term memory (TDLSTM) neural network to predict the software faults of the cloud VM.
- Third, we formulate the cloud service reliability and cost aware job scheduling as a linear programming problem. To reduce job service cost and rejection rate, we introduce a scheduling cost that integrated a response time slack factor to evaluate the optimal scheduling service and VM pair.
- Fourth, a low complexity heuristic greedy RCJS algorithm is proposed to optimize the service scheduling cost. The algorithm tries to minimize the cloud service cost, rejection rate, and improve the service execution reliability.
- Finally, we describe the results of the performance evaluation on two simulation experimental platforms; the results clearly show that our proposed RCJS algorithm significantly outperforms OPVMP and MIN-MIN algorithms in terms of average service cost and rejection rate.

The organization of this paper is as follows: Section 2 reviews related work. In Section 3, we provide a systems resources management architecture. The service execution reliability model is designed in Section 4. Section 5 proposes a heuristic greedy reliability and cost aware job scheduling algorithm. In Section 6, we evaluate our proposed RCJS algorithm and analyze the experimental results. Finally, we conclude this study and offer some comments on further research in Section 7.

2 Related Work

Recently, cloud users as well as cloud providers are paying increasing attention to the cloud computing systems Quality of Service (QoS), such as energy consumption, security, cost, response time, resource utilization, and reliability [3], [12], [13], [14], [20], [36], [37]. These QoS issues have an important impact on cloud computing systems performance. In the paper [36], the authors proposed multi-criteria decision making models to rank cloud services multiple QoS attributes for systems selecting. In most case, we only focus on some of these QoS attributes. G. L. Stavrinides and H. D. Karatza proposed an energy-efficient, QoS-aware, and cost-effective scheduling strategy, which aimed at reducing the cost of cloud users in an energy-efficient method for the cloud computing systems providers [3]. In the paper[12], the authors introduced an approach to pack cloud requests of the same service type into the same VM, which can improve resource utilization and power consumption with guaranteed QoS. G. Xie *et al.* focused on the time consumption and reliability QoS-aware task scheduling in cloud systems [13]. In paper [37], for the response time QoS constraint, S. Long *et al.* proposed a best response algorithm (BRA) to optimize total cost on the collaborative edge and cloud systems. P.K. Sahoo *et al.* designed a Link based Virtual Resource Management (LVRM) algorithm to map the VMs onto physical machines based on the available and required resources [32]. H. Xu *et al.* focused on the malicious attacks and security protection mechanism to improve cloud systems security QoS. These research results reveal that QoS-Aware approaches can effectively improve the performance of cloud computing systems and satisfy cloud users various requirements. Therefore, it is a meaningful research work to improve QoS of cloud services.

As an important part of the QoS of a cloud computing system, cloud service reliability is also of concern to the academia and industry [5]. Checkpointing is one of service reliability enhancement approaches, which periodically saves the execution state as an image file [15]. However, owing to the large scale of the VMs in cloud computing systems, this approach consumes a large storage space. Another effective reliability assurance mechanism is the fault-tolerant technique based on replication, which attempts to map each primary VM to one or more backup VMs [5], [10], [11], [16]. In paper [10], the authors proposed an optimum VM placement approach according to the performance requirements of the application services; moreover, this method was tolerant to k number of host server failures. X. Zhu *et al.* extended the traditional PB technique based on cloud characteristics and built a real-time workflow fault-tolerant model [16]. In a variation from the above PB approach, A. Zhou *et al.* proposed an OPVMP strategy to enhance cloud service execution reliability with k-fault-tolerance assurance [5]. These fault-tolerant strategies without considering reliability will inevitably result in higher computational cost. In fact, most cloud services can be successfully completed owing to the high reliability of the actual cloud computing systems, and only a few of them need fault-tolerance. In contrast, our proposed scheduling strategy attempts to analyze and predict the execution reliability of cloud services, and only backups low-reliability services. Therefore, we only need to backup a few cloud services, so as to improve the reliability of services and reduce the computing cost of the whole system.

A good method to achieve them is to analyze and predict the execution reliability of cloud services and assign a backup VM to the low-reliability services. Computing systems reliability analysis, especially of physical resources, is a classical research topic [6], [17], [18]. A. Dogan and F. *Özgü*ner formally analyzed the reliability of heterogeneous computing systems and proposed a reliability-aware dynamic level scheduling algorithm [17]. This research provides an important research direction for the cloud services execution reliability analysis on physical resources. Z. Wen *et al.* use exponential distribution to build workflow reliability model on federated clouds [26]. In our previous work [18], we adopted Weibull distribution to build the heterogeneous cluster application reliability analysis model. X. Li *et al.* used the queuing and graph theories to build a service reliability model on a cloud data center [6]. H. Balla *et al.* proposed a reliability enhancement resource allocation strategy, which can effectively adapt to the dynamic changes in cloud systems [19]. However, the above strategies mainly focused on the physical resources analysis, and did not consider VM software fault. This work focuses on not only the reliability of VM physical resources, but also the VM software faults.

The cloud QoS prediction is an important work and helpful for systems to schedule services to the most suitable VMs. The classical approach is resource reservation, which can enable efficient task scheduling [39]. In the paper [7], [21], the authors used the traditional multivariate or average-based time series analysis method to predict cloud QoS requirements or VM failure. X. Chen *et al.* proposed an iterative cloud QoS prediction model based on the system historical data [40].

In recent years, machine learning techniques are widely used in cloud QoS prediction. W. Li *et al.* proposed a Bayesian network model to predict cloud QoS based on the correlation between the interactive information of the hardware/software resources [35]. This Bayesian network prediction reasoning algorithm can accurately predict cloud services future QoS values. A similar way, the authors use the machine learning regression based on historical load QoS characteristics to predict cloud systems resource levels and scale components [38]. Certain artificial intelligence methods are also used in computing systems prediction, such as TDLSTM neural network workload prediction [22] and resource prediction in data centers based on machine learning [23]. These approaches have achieved good analysis and prediction results. However, the machine learning schemes require plenty training data, which in many cases requires a lot of time. In this work, we use parallel techniques to reduce model training and prediction time.

Therefore, to address the cloud service reliability issue, we first adopted Poisson distribution to analyze the failures of the physical resources of cloud VMs. Then, we utilized the TDLSTM neural network to predict the software fault of the cloud VM. The difference between related work [22] and this study is that we adopt a CUDA-enabled accelerated parallel TDLSTM to reduce the prediction time on Multiple VMs. Finally, we proposed a heuristic greedy RCJS algorithm that satisfies the cloud service response time.

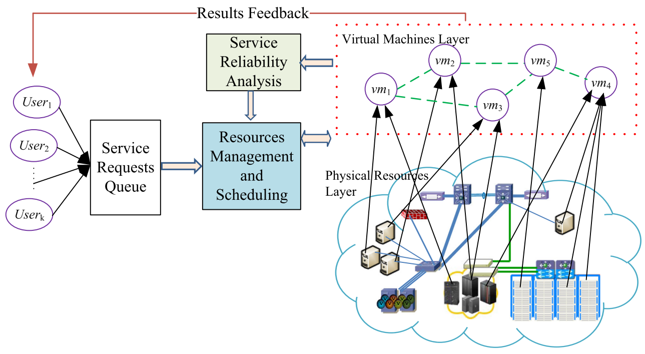


Fig. 1. Cloud resources management architecture.

3 Cloud Resources Management Architecture

Fig. 1 depicts the resources management architecture in cloud computing systems. The model is primarily composed of users, service requests queue, service reliability analysis, resource management and scheduling, VMs layer, and physical resources layer. This system is responsible for processing the service requests and providing the computing results to users. The following service reliability analysis and scheduling strategy are supported by this architecture. In this section, we first define the notations of cloud service requests. We then provide cloud virtualization layer and the main characteristics of virtual machines. We also discuss the definition of cloud service reliability and the causes of service failures.

For ease of understanding, we summarize the main notations and their definitions used throughout of this paper in Table 1.

3.1 Cloud Computing Service Requests

Many cloud computing systems provide Software as a Service (SaaS) or Service-Oriented Architecture (SOA) technologies to handle the numerous concurrent service requests of users [24], [34]. These cloud service requests are submitted by users and inserted into the system requests queue. Then, these service requests are scheduled to VMs by the resource management and scheduling module according to the scheduling metrics, such as completion time, reliability, and cost. The VMs can be constructed dynamically according to user service request characteristics based on virtualization technology.

In real-world life, many cloud service requests are independent, and there are also many interdependent cloud service requests. However, with the development of Internet of things and 5g mobile communication technology, a large number of micro-service requests are submitted to cloud computing system, and most of them are independent of each other. Therefore, in this paper, the cloud service requests are assumed to be independent and denoted as a

TABLE 1
The Notations and Definitions

Notation	Definition
s_i	the ith cloud service
$w(s_i)$	the request size of cloud service s_i
$ss(s_i)$	the storage required for service s_i
$ar(s_i)$	the service s_i request arriving time
$rt(s_i)$	the service s_i response time
vm_j	the jth cloud systems virtual machine
$w(vm_j)$	the computation capacity of the VM vm_j
$st(s_i, vm_j)$	the processing time of cloud service s_i on VM vm_j
$ES(s_i, vm_j)$	the start execution time
$RT(s_i, vm_j)$	the running time
PR_{s_i,vm_j}	the physical resources reliability of the cloud service
SR_{s_i,vm_j}	the software fault prediction value
$c(s_i, vm_j)$	the cost of cloud service s_i on VM vm_j
$c(S)$	total cloud service cost
$rst(s_i, vm_j)$	the response slack time
$Scost(s_i)$	the service s_i scheduling cost

set of services $S = \{s_1, s_2, \ldots, s_n\}$. The notation $w(s_i)$ represents the request size of cloud service s_i; $ss(s_i)$(G) is the storage required for service s_i and G denotes Gigabyte; $ar(s_i)$(s) is the arriving time for the user to submit the service request and s denotes second. In general, the cloud users expect their service requests to be successfully satisfied in a reasonable time, which we call response time. Here, the response time $rt(s_i)$ represents the interval between a user submitting his service request and the correct feedback being received.

3.2 Cloud Virtual Machines (VMs)

Generally, the physical resources layer of cloud computing systems consists of a set of physical servers, storage, networks, and other devices. These resources are heterogeneous, geographically distributed or centralized, as shown in the physical resources layer in Fig. 1. The cloud virtualization technologies, such as VMware, Xen, KVM, allow the physical resources of the systems to be dynamically separated into multiple VMs or collaboratively reconstructed into a single VM.

In this study, the cloud virtualization layer consists of a pool of VMs, which is represented by $\Omega = \{vm_1, vm_2, \ldots, vm_m\}$. Each cloud VM, $vm_j \in \Omega$, is associated with different parameters, such as the computation capacity, computation cost, storage cost, resource reliability assurance rate. The computation capacity of the VM is measured in million instructions per second and denoted as $w(vm_j)$. Here, the processing time $st(s_i, vm_j)$ of cloud service s_i on VM vm_j is defined as

$$st(s_i, vm_j) = \frac{w(s_i)}{w(vm_j)}. \tag{1}$$

Where the processing time is service request size $w(s_i)$ divided by cloud VM computation capacity $w(vm_j)$, and measured in seconds.

The computation cost $cc(vm_j)$(\$) is the cost per hour on VM vm_j; the storage cost $sc(vm_j)$(\$) denotes the cost per G of cloud VM storage; the service delay by VM vm_j is denoted as $sd(vm_j)$(s), and the running time of vm_j is expressed as $runTime(vm_j)$(s). The symbol $prTr(vm_j)$ indicates the physical resources reliability assurance rate of the VM. When the reliability of cloud service on VM vm_j is less than the threshold $prTr(vm_j)$, in our solution, another backup VM performs this service to improve its reliability. Table 2 lists certain characteristics of cloud VMs as an example.

TABLE 2
The Cloud VM Characteristics

VM	w(MIPS)	cc(\$)	sc(\$)	$sd(s)$	$prTr$	$runTime(s)$
VM_1	4500	0.65	0.78	0.03	0.96	3450
VM_2	7350	0.82	1.2	0.14	0.97	234234
VM_3	5600	0.77	1.02	0.012	0.84	345
VM_4	16500	1.83	0.98	0.19	1.89	9856

3.3 Service Reliability

The cloud service reliability is the ability of cloud computing systems to provide the correct feedback successfully to cloud user requests under the stated conditions in an acceptable time [6]. This reliability is an important component of cloud QoS; moreover, it is regarded as a key concern for both users and cloud computing providers. In reality, many cloud services fail to fulfill their service assurance owing to cloud physical resources failures, VM software execution faults, and network communication delay. In our architecture, the service reliability analysis module is first used to compute VM service reliability. Then, the resource management and scheduling module attempts to improve the cloud systems QoS, such as service reliability.

4 Service Reliability Analysis

The reliability of service execution on VMs is one of the most complex issues in cloud systems. This section first provides the cloud VM physical resources reliability calculation method. Then, we analyse cloud VM software execution failures and use the TDLSTM neural network to predict software faults. Due to the large-scale of neural network training computation for software fault prediction, we propose a parallel computing method based on CUDA.

4.1 Cloud Physical Resources Failures

The physical resources of cloud computing systems are heterogeneous; moreover, most of them are based on a very-large-scale integration chip. These types of high-density circuits are susceptible in causing instantaneous faults owing to circuit crosstalk and delay. Second, high integrated circuits are prone to hot carrier degradation, electron migration, and gate oxygen breakdown, which results in permanent failure. Therefore, the physical resources failure occurs with wear, aging, random failure, and life failure [25]. In our model, those failures are assumed to follow a Poisson process [17], [26].

The VM vm_i of the cloud computing systems is primarily composed of a set of A physical servers, where $PS = \{ps_1, ps_2, \ldots, ps_A\}$, storage pt, and networks pn. We assume that each resource is associated with a constant failure rate λ, such that the failure rate of the physical server, storage, and networks are λ_{ps_a}, λ_{pt}, and λ_{pn}, respectively [17], [26].

Here, the exponential probability density function f of a physical resource, such as a physical server λ_{ps_a}, is $f_{ps_a}(t) = \lambda_{ps_a} e^{-\lambda_{ps_a} t}$. Therefore, the reliability function $P_{ps_a}(t)$ of this resource can be expressed as [17]

$$\begin{aligned} P_{ps_a}(t) &= 1 - \int_0^t f_{ps_a}(t) dt \\ &= e^{-\lambda_{ps_a} t}. \end{aligned} \tag{2}$$

For the cloud service s_i that is processed by VM vm_j, the start execution time $ES(s_i, vm_j)$ is the latest time of cloud service arriving time $ar(s_i)$ and the available time $avail(vm_j)$ of virtual machine, which is expressed as

$$ES(s_i, vm_j) = Max\{ar(s_i), avail(vm_j)\}. \tag{3}$$

Therefore, for service s_i on VM vm_j, the running time (s) is the sum of start execution time, service processing time, and VM running time, which can be defined by

$$RT(s_i, vm_j) = ES(s_i, vm_j) + st(s_i, vm_j) + runTime(vm_j). \tag{4}$$

The physical resources reliability of the cloud service, PR_{s_i,vm_j}, has the following corollary

Corollary 1. *The physical resources reliability PR_{s_i,vm_j} of cloud service s_i on VM vm_j is*

$$PR_{s_i,vm_j}[RT(s_i, vm_j)] = e^{-(\Sigma_{a=1}^{A} \lambda_{ps_a} + \lambda_{pt} + \lambda_{pn}) RT(s_i, vm_j)}. \tag{5}$$

Here, A is the number of physical resources.

Proof. For each physical resource of VM vm_j, such as physical server $ps_a, a \in \{1, 2, 3, \ldots, A\}$, the reliability for cloud service s_i is

$$P_{s_i,ps_a}[RT(s_i, vm_j)] = e^{-\lambda_{ps_a} RT(s_i, vm_j)}.$$

The integration of multibillion transistors in heterogeneous physical resources failures are independent of each other, and therefore their reliability PR_{s_i,vm_j} can be expressed as

$$\begin{aligned} PR_{s_i,vm_j}[RT(s_i, vm_j)] &= \Pi_{a=1}^{A} P_{s_i,ps_a}[RT(s_i, vm_j)] \\ &\quad \times P_{s_i,pt}[RT(s_i, vm_j)] \times P_{s_i,pn}[RT(s_i, vm_j)] \\ &= e^{-\Sigma_{a=1}^{A} \lambda_{ps_a} RT(s_i, vm_j)} \times e^{-\lambda_{pt} RT(s_i, vm_j)} \\ &\quad \times e^{-\lambda_{pn} RT(s_i, vm_j)} \\ &= e^{-(\Sigma_{a=1}^{A} \lambda_{ps_a} + \lambda_{pt} + \lambda_{pn}) RT(s_i, vm_j)}. \end{aligned}$$

□

4.2 Cloud VM Software Fault Prediction

The software execution reliability ensures the provision of the right services under specified operating conditions [25]. Unlike the physical resources reliability in the cloud, the software faults are caused by issues in the design, coding, and inherent logic associated with user service requests. Essentially, the VM software execution failures are primarily due to the changes of cloud service, system maintenance, and cloud computing environment over time. Many research work, such as paper [7], [30], [31], have shown that the software execution faults are time series event. Moreover, most cloud service requests have the characteristic of random and two-dimensional time series. For example, online office requests to a high-frequency cloud occur between the times of $8:10$ and $11:50$ during the working day, and the number of cloud service requests on weekends is relatively small. In general, the higher the system workload, the higher the probability of software faults. Therefore, due to the day and time two-dimensional characteristics of the cloud system workload, the software fault occurrence in a cloud VM is also two-dimensional.

TABLE 3
The Example of Time Series Software Execution Fault

Day/time	10:00	10:30	11:00	11:30	12:00	12:30	13:00	13:30
4-6-2019	1	1	1	1	1	1	1	1
4-7-2019	1	0	1	1	1	1	1	1
4-8-2019	1	1	1	1	1	1	1	1
4-9-2019	1	1	0	1	1	1	1	1

In this paper, we use the day and time-based two-dimensional discrete time series with an interval of 30 min to represent cloud software execution faults. Table 3 lists an example of this two-dimensional discrete time series software execution fault from 04-06-2019 to 04-09-2019 between 10:00 and 14:00. Here, we assume that $x^{d,t}$ indicates the software execution fault according to the day d and time t. Further, we let $x^{d,t} = 0$ indicate that the VM has a fault and cannot effectively provide cloud services. Otherwise, $x^{d,t}$ is set as 1.

In our solution, we used the TDLSTM neural network adopted in [22] to predict the future software execution fault $x^{d,t+1}$ of the cloud VM. This neural network model input data X^t can be defined as

$$X^t = [x^{d-b,t}, \ldots, x^{d,t}] \Lambda [x^{d,t-g}, \ldots, x^{d,t}], \tag{6}$$

where X^t is a binary string with $b + g + 2$ bits, which combines the previous $b + 1$ days at time t and $g + 1$ software execution status time series. These data are used in the TDLSTM neural network cells as input, and the input gate, forget gate, output gate, and the memory are achieved using the following functions [22]:

$$I^t = W^{(ig)} X^t + h^{(d-1,t)} UD^{(ig)} + h^{(d,t-1)} UH^{(ig)}. \tag{7}$$

$$F^t = W^{(fg)} X^t + h^{(d-1,t)} UD^{(fg)} + h^{(d,t-1)} UH^{(fg)}. \tag{8}$$

$$O^t = W^{(og)} X^t + h^{(d-1,t)} UD^{(og)} + h^{(d,t-1)} UH^{(og)}. \tag{9}$$

$$G^t = W^{(cm)} X^t + h^{(d-1,t)} UD^{(cm)} + h^{(d,t-1)} UH^{(cm)}. \tag{10}$$

$$C^t = \sigma(I^t) * tanh(G^t) + \sigma(F^t) * C^{t-1}. \tag{11}$$

$$h^t = \sigma(O^t) * tanh(C^t). \tag{12}$$

Here, the notations ig, fg, og, and cm denote the input gate, forget gate, output gate, and memory, respectively. The output layer produces a prediction value y^t, which is expressed as

$$y^t = \sigma(W^{(out)} h^t). \tag{13}$$

Further details regarding this TDLSTM neural network model can be obtained from our previous paper [22].

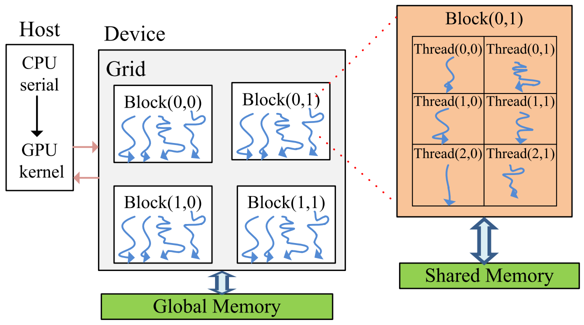


Fig. 2. The GPU block-thread and memory architecture.

Therefore, for service s_i on cloud VM vm_j with an execution interval $[runTime(vm_j)+ES(s_i,vm_j), RT(s_i,vm_j)]$, we can use the TDLSTM neural networks to predict software fault.

4.3 CUDA-Enabled Parallel TDLSTM Neural Network

Cloud computing systems may have dozens, hundreds, or even thousands of VMs. Each TDLSTM neural network for software fault prediction in a VM is associated with a series of training data. Thus, a significant amount of time is required to train the TDLSTM prediction model. An efficient technique is to adopt a GPU for the parallel training of the TDLSTM neural network model.

In this study, we used the CUDA framework released by NVIDIA Corporation to implement a parallel TDLSTM. The NVIDIA GPU Device consists of a lot of Thread Blocks, and each Block is also composed of many Threads [28]. The communications among the threads of a Block threads are achieved through shared memory. However, the communication between Thread Blocks can only be achieved through global memory, which is lower than the shared memory in communication efficiency. Fig. 2 depicts this GPU Block-Thread and Memory architecture. Therefore, our parallel strategy is to provide a GPU Thread Block to each VM TDLSTM prediction model.

In our solution, the data of the software faults history of the cloud VM are stored in the corresponding VM files, which is stored into a day and time-based two-dimensional array in the GPU shared memory. Moreover, the input and hide layer weight parameter matrix are essentially floating point representations and stored in the GPU shared memory in our implementation. For each training, the main calculation is located in the input gate, forget gate, output gate, and memory of the hidden layer. From these observations, we assigned each GPU thread a calculation of the hidden layer node parameters, and all nodes could be parallel computed in a GPU block. Thus, the TDLSTM neural network model training is implemented with CUDA-enabled block and thread level parallel computations.

5 Cloud Service Scheduling Strategy

This section proposes a heuristic reliability and cost aware job scheduling (RCJS) algorithm for cloud service requests that aims at improving cloud service processing reliability with minimum cost. We first outline the total cloud service cost, the service rejection rate, and formulate them as a linear programming problem. Then, we introduce a scheduling cost that integrated a response time slack factor. Third, we propose a heuristic RCJS algorithm to solve this cost and reliability optimization problem. Finally, the time complexity of RCJS scheduling strategy is discussed.

5.1 Problem Definition

In this study, the service requests, also called as jobs, are submitted to cloud computing systems for processing with cost, which is paid by the user. We consider $X_{i,j}=1$ as the service s_i that is scheduled on the jth VM vm_j; otherwise, we consider $X_{i,j}=0$. Here, the cost $c(s_i,vm_j)$ (\$) of cloud service s_i on VM vm_j is the sum of computation cost and storage cost, and can be expressed as

$$c(s_i,vm_j)=X_{i,j}\bigg[\frac{st(s_i,vm_j)}{3600}\times cc(vm_j) + ss(s_i)\times sc(vm_j)\bigg]. \tag{14}$$

Our cloud service scheduling strategy uses the PB approach to improve service reliability. Therefore, for service s_i scheduled on VM vm_j, we consider a case where cloud service physical resources reliability PR_{s_i,vm_j} and software fault prediction SR_{s_i,vm_j} satisfy the following equations:

$$\begin{cases} PR_{s_i,vm_j} \le prTr(vm_j)\ or, \\ SR_{s_i,vm_j}=0. \end{cases} \tag{15}$$

In this case, we schedule the cloud service s_i to another backup VM, such as vm_k, and $X_{i,k}$ is also set as $X_{i,k}=1$. Conversely, we do not require another VM to execute service s_i. Therefore, the cost $c(s_i)$(\$) of the cloud service s_i is the total of the primary and backup execution costs, which is expressed as

$$\begin{cases} c(s_i)=\sum_{vm_j\in\Omega} c(s_i,vm_j), \\ 1\le \sum_{1\le j\le m} X_{i,j} \le 2 \quad \forall\ i\in[1,2,\ldots,n]. \end{cases} \tag{16}$$

Where, the constraints of Eq. (16) mean that the cloud service is scheduled at less than one VM and no more than two VMs. Therefore, the total cloud service cost $c(S)$ is the sum of all service processing cost and defined as

$$c(S)=\sum_{s_i\in S} c(s_i). \tag{17}$$

The other scheduling algorithm constraint is the response time, which follows the following inequality:

$$ES(s_i,vm_j)+st(s_i,vm_j)+sd(vm_j)\le ar(s_i)+rt(s_i). \tag{18}$$

From this inequality, we can conclude that the cloud service s_i may be rejected by the system because it does not satisfy the service response time; moreover, $X_{i,j}=0\ \forall\ i\in[1,2,\ldots,n]$. We define the cloud service rejection rate SRR as

$$SRR=\frac{SRN}{n}. \tag{19}$$

Where SRN is the number of cloud services that failed to schedule successfully, and $SRN=\sum s_i \quad if\ X_{i,j}=0\ \forall j$.

Therefore, the cloud service rejection rate SRR is the ratio of SRN to total services.

The objective of our proposed cloud service scheduling algorithm is to minimize the total cloud service cost and rejection rate, which can be formulated as the following linear programming problem:

$$Minimize\ c(S) \tag{20a}$$

$$Minimize\ SRR. \tag{20b}$$

And the restraint condition is

$$\begin{cases} s.t. \\ X_{i,j} = 1\ Or\ X_{i,j} = 0, \\ 0 \le \sum_{1 \le j \le m} X_{i,j} \le 2 \quad \forall\ i \in [1, 2, \ldots, n]. \end{cases}$$

5.2 Heuristic Greedy Reliability and Cost Aware Job Scheduling Algorithm

In the cloud computing platform, all service requests must be completed within a reasonable time $rt(s_i)$, which we call the response time. Otherwise, the service will be rejected by cloud system, and the corresponding system will be also considered to be unreliable. Therefore, the service with less response slack time should be scheduled first. In our solution, we integrated a response time slack factor α into the cloud service cost computing method. We first define the response slack time $rst(s_i, vm_j)$, which is the interval between response time and service completion time. This is expressed as

$$\begin{aligned} rst(s_i, vm_j) = {} & ar(s_i) + rt(s_i) \\ & - [ES(s_i, vm_j) + st(s_i, vm_j) + sd(vm_j)]. \end{aligned} \tag{21}$$

In this algorithm, we introduce a scheduling cost $Scost(s_i, vm_j)$ to evaluate the optimal scheduling service and VM pair, which is defined as

$$\begin{aligned} Scost(s_i, vm_j) = {} & \alpha \times \left[\frac{st(s_i, vm_j)}{3600} \times cc(vm_j) \right. \\ & \left. + ss(s_i) \times sc(vm_j) \right] \\ & + (1 - \alpha) \times \left[\frac{rst(s_i, vm_j)}{3600} \times cc(vm_j) \right]. \end{aligned} \tag{22}$$

From Eq. (22), for a service on a VM, we can conclude that the less response slack time must has less service cost, this service is more likely to be scheduled at the earliest possible time. Thus, our proposed RCJS algorithm can decrease job rejection rate and result in high reliability.

Accordingly, the service s_i scheduling cost $Scost(s_i)$ is also the total of its primary and backup scheduling costs and is expressed as

$$\begin{cases} Scost(s_i) = \sum_{vm_j \in \Omega} Scost(s_i, vm_j), \\ 1 \le \sum_{1 \le j \le m} X_{i,j} \le 2 \quad \forall\ i \in [1, 2, \ldots, n]. \end{cases} \tag{23}$$

The pseudo code of the heuristic greedy RCJS algorithm is presented in *Algorithm 1*. Initially, we initialize the parameters of the VMs in the cloud system, such as computation capacity, computation cost, storage cost, delay, reliability threshold, running time, and available time. Then, the CUDA-enabled parallel TDLSTM neural network is used to train the software fault prediction model of the VM. The following steps are utilized to identify optimal job-VM matching pairs:

Algorithm 1. The RCJS Algorithm

```
Input: Cloud service requests set $S$
Output: An assignment $X_{i,j}$
1  Cloud VMs Initialization;
2  Use CUDA-enabled parallel TDLSTM to train VM software
   fault prediction model;
3  while there are unscheduled jobs do
4    for each unscheduled job $s_i$ do
5      for each VM $vm_j$ and satisfy Eq. (18) do
6        Compute running time in Eq. (4);
7        Compute physical resources reliability $PR_{s_i,vm_j}$ in
         Eq. (5);
8        Use TDLSTM to predict $vm_j$ software fault $SR_{s_i,vm_j}$;
9        if satisfy Eq. (15) then
10         for each other VM $vm_k$ and satisfy Eq. (18) do
11           Use Eq. (14) to compute cost $c(s_i, vm_k)$;
12         end
13         Find a VM $vm_k$ with minimum cost;
14         Let VM $vm_k$ as a backup VM;
15       end
16       Use Eqs. (22) and (23) to compute scheduling cost
         $Scost(s_i)$ on primary/ backup VM;
17     end
18     if there are scheduling solutions for job $s_i$ then
19       Find the primary/backup VMs for job $s_i$ with mini-
         mum scheduling cost;
20     else
21       Reject job $s_i$;
22     end
23   end
24   Find the job $s_i$ and primary/backup VMs with minimum
     scheduling cost;
25   Use Eq. (16) to compute job cost $c(s_i)$;
26   Assign job $s_i$ to primary/backup VM;
27   Update VMs available time;
28   Remove job $s_i$ from unscheduled job set;
29 end
```

Steps 3 to 5. We attempt to find the optimal solution from all job-VM pairs. The selected VM must satisfy the service s_i on VM vm_j such that the execution time is less than the response time of service s_i.

Steps 6 to 8. Initially, we compute service s_i on VM vm_j by running Eq. (4), which is applied to compute the physical resources reliability probability PR_{s_i,vm_j} when service s_i is being executed. We also predict the VM software fault SR_{s_i,vm_j} in this time interval.

Steps 9 to 15. We attempt to backup services that demonstrate a low physical resources reliability PR_{s_i,vm_j} or where the software fault prediction SR_{s_i,vm_j} is zero. In these cases, we attempt to identify other backup VMs with minimum cost.

Steps 16. In this step, we use Eqs. (22) and (23) to compute the scheduling cost of service s_i on its VMs.

Steps 18 to 22. For a cloud service, we try to find job and VMs pair with minimum scheduling cost. If there is not a scheduling solution, the algorithm will reject this job.

Steps 24 to 28. These steps attempt to determine the optimal job-VM pair with minimum scheduling cost for all unscheduled cloud services. Then, the algorithm assigns job to the corresponding VM and updates all VMs and unscheduled job set.

5.3 Time Complexity

The time complexity of this RCJS strategy can be expressed in terms of the number of services n, number of VMs in the cloud system m, and maximum physical resources k of the VMs. The time complexity of this RCJS algorithm is analyzed as follows. All parameters of the VMs in the cloud systems are initialized in Step 1 of the algorithm as $O(m)$. Step 7 attempts to compute the physical resources reliability PR_{s_i,vm_j}, which can be performed in a time $O(k)$. Steps 9 to 15 determine other backup VMs with time complexity $O(m)$. Generally, the maximum number of physical resources k of the VMs is considerably smaller than the number of VMs. Therefore, the time complexity in Steps 6 to 17 is $O(m)$. In our proposed RCJS algorithm, there are three loops in Steps 3 to 5, which can be performed in time mn^2. Thus, the overall time complexity of the RCJS algorithm is $O(m^2n^2)$.

6 Performance Evaluation

In the following sections, our proposed heuristic greedy RCJS algorithm is compared with two existing approaches: OPVMP [5] and MIN-MIN [28]. The OPVMP attempted to improve the cloud service execution reliability based on a k-fault-tolerant model. The MIN-MIN algorithm attempted to schedule a job with minimum job execution finish time. The primary performance metrics chosen for the performance evaluation were the average cost, schedule length, average reliability, and rejection rate (see Eq. (19)). The average cost $acost$ and average reliability $areb$ are the average values cost and reliability of all successfully completed cloud services, respectively, which are defined by

$$\begin{cases} acost = \frac{c(S)}{n-SRN}, \\ areb = \frac{\sum_{s_i\in S}(PR_{s_i} \ or \ SR_{s_i})}{n-SRN}, \end{cases} \tag{24}$$

where, SRN is the number of rejected cloud services. The schedule length is the latest service finish time.

In this section, we first build an extensive simulation experimental platform. Then, we test the sensibility of slack factor α with 50, 300, 800 jobs. Finally, we compare our proposed RCJS with two existing algorithms.

6.1 Experimental Platform

To test the performance of our proposed RCJS algorithm, we built an extensive discrete event simulator, which implemented in java programming language using the Cloud Simulation toolkit (CloudSim)[33]. The first test cloud platform was a randomly generated framework with the number of VMs ranging from 160 to 260 in steps of 10. Each VM computation capacity was generated randomly from a uniform distribution within the range of [3000, 50000]. The number of physical resources of the cloud VM was approximately determined in proportion to the VM computation capacity; further, the VM physical resources failure rate λ was assumed to be uniformly distributed between 1×10^{-3} and 1×10^{-5} failures/hour. A part of the historical data of the cloud VM software faults was retrieved from yeCloud, and the other fault dataset was downloaded from Failure Trace Archive (FTA) [29].

The second experimental cloud environment originated from real-world platforms [34], which consisted of VMs with various physical resources, operating systems, and software. Specifically, this cloud platform consists of the following:

- Ten VMs, each VM with a computation capacity of 10000 at 0.75\$/hr, two physical resources with failure rate: $\lambda = 0.00002$, 4 GB RAM, 1000 GB disk failure rate: $\lambda = 0.0001$ at 0.23\$/G, network failure rate: $\lambda = 0.00005$.
- Five VMs, each VM with a computation capacity of 25000 at 0.98\$/hr, one physical resource with failure rate: $\lambda = 0.00001$, 2 GB RAM, 1000 GB disk failure rate: $\lambda = 0.00015$ at 0.45\$/G, network failure rate: $\lambda = 0.00003$.
- Three VMs, each VM with a computation capacity of 8000 at 0.3\$/hr, one physical resource with failure rate: $\lambda = 0.0001$, 6 GB RAM, 2000 GB disk failure rate: $\lambda = 0.00008$ at 0.15\$/G, network failure rate: $\lambda = 0.0002$.
- Two VMs, each VM with a computation capacity of 40000 at 1.1\$/hr, five physical resources with failure rate: $\lambda = 0.00001$, 10 GB RAM, 500 GB disk failure rate: $\lambda = 0.00052$ at 0.32\$/G, network failure rate: $\lambda = 0.00001$.

The cloud services requirements were generated according to cloud computing systems user submitted information, such as Amazon EC2 [34]. The service request size $w(s_i)$ is a random multiple of the average computing capacity of the system VMs. The service storage requirement $ss(s_i)$ is the uniform distribution within [0.05, 10]G.

6.2 Response Time Slack Factor Sensibility

In our proposed RCJS algorithm, the response time slack factor α is a significant influencing factor. Therefore, to test the factor α, we conducted a comparison on the first test cloud platform with 200 VMs. The processing capacity of the cloud services (or jobs) was randomly generated from a uniform distribution [100, 5000] of VMs with average computation capacity. The services storage, arriving time, response time were also randomly generated. In this experimental environment, we varied the response time slack factor α from 0.1 to 1 in steps of 0.1. The larger the slack factor α value, the scheduling metric is more dependent on the cloud service cost. According to the system with 200 VMS, we choose 50, 300, 800 jobs for low, medium, and high workload, respectively.

Figs. 3, 4, and 5 show the simulation experimental results with 50, 300, and 800 jobs, respectively. For the average cost shown in Figs. 3a, 4a, and 5a, the unit of these Figs. are dollor(\$); For the schedule length shown in Figs. 3b, 4b, and 5b, the unit of these Figs. are second(s). These unit are also applied to the later Fig. 6 and 7. For the low workload of 50 jobs, the average cost and rejection rate are almost the same.

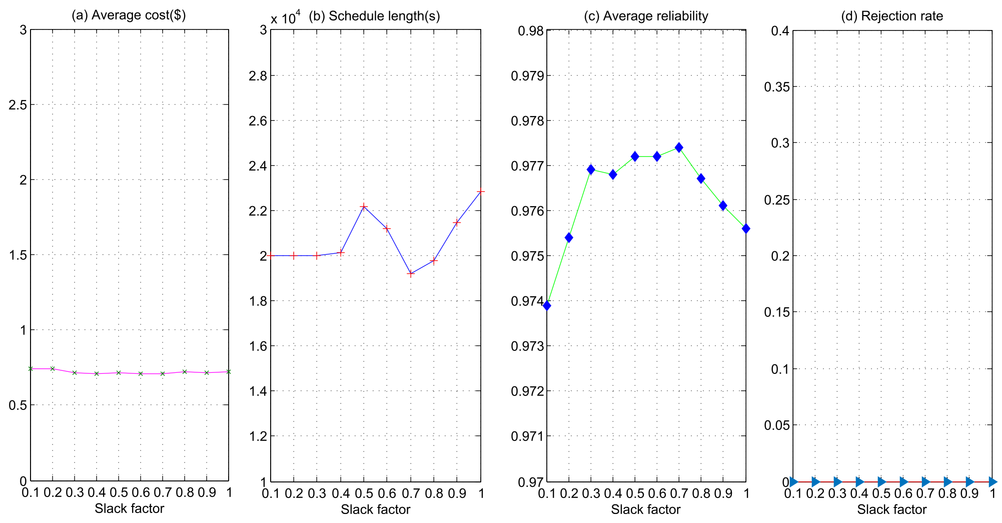


Fig. 3. Response time slack factor α with 50 jobs.

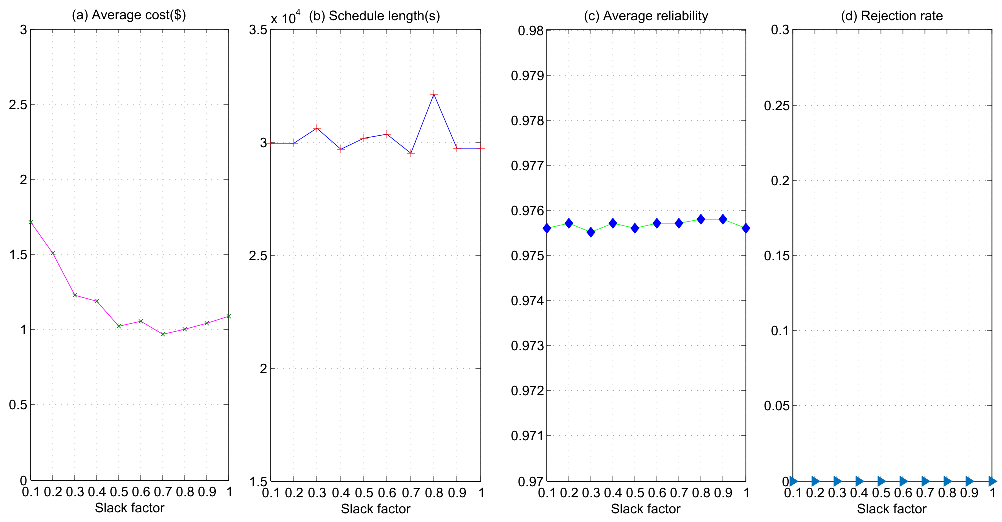


Fig. 4. Response time slack factor α with 300 jobs.

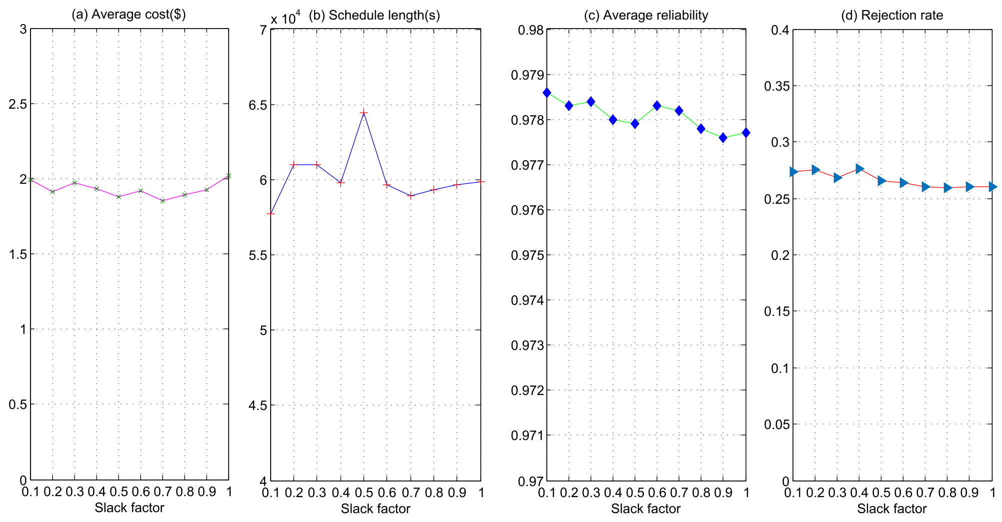


Fig. 5. Response time slack factor α with 800 jobs.

The optimal schedule length and average reliability are almost at slack factor value of 0.7 in Figs. 3b and 3c, respectively. From Fig. 4, we can also conclude that the optimal average cost and schedule length are at slack factor α with 0.7, and the rest are basically the same. Fig. 5 are the results of high workload with 800 jobs, which indicates that the optimal average cost, schedule length, and average reliability are also at slack factor $\alpha = 0.7$. Due to the high workload,

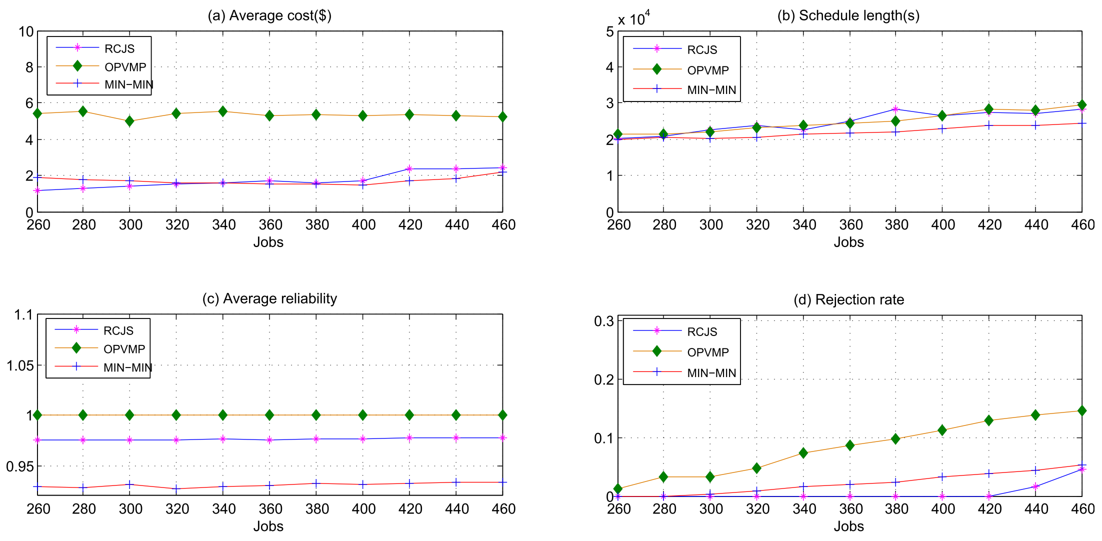


Fig. 6. The comparison results of various jobs on 200 VMs cloud platform.

the rejection rate here is also relatively high, but decreases with the slack factor α increases. Thus, we believe that the value $\alpha = 0.7$ is the best response time slack factor for our proposed RCJS algorithm, which is used in the following experiments.

6.3 Comparison Results

The goal of these experiments was to compare the proposed RCJS algorithm with randomly generated cloud services on the first test platform. Table 4 lists the scheduling results of the three comparison algorithms for 400 jobs on 200 VMs, in which we have randomly listed the serviceIDs from 240 to 259. From this table, we can conclude that most of the jobs scheduled by our proposed RCJS algorithm are 1 VM, and while only 2 jobs are scheduled on 2 VMs. On the other hand, jobs scheduled by OPVMP algorithm is at least on 2 VMs, and many jobs are scheduled on 3 or even 4 VMs. Therefore, the cost and reliability of the OPVMP algorithm are all higher than that of RCJS and MIN-MIN algorithms.

There are also some special phenomena, such as serviceID 247, the cost of RCJS is $1.293\$$ and that of OPVMP is $16.4\$$. In this case, the OPVMP assign VM 125 as its primary VM, the execution time is 2691s, and the execution cost on the primary VM 125 is only $1.127\$$. However, the OPVMP assign this service a backup VM 140, the execution time is 5132s and is larger than that of the primary VM, the execution cost is $15.273\$$, for the VM 140 computation cost $cc(vm_{140})$ is $10.7\$$ per hour. Therefore, the service cost produced by OPVMP is $16.4\$$, which is much larger than that of RCJS. This is due to the fact that our proposed RCJS algorithm tries to find the optimal VM with the minimum service cost. The minimum service cost may be much less than the maximum cost of such service on other high cost VM. In some actual cloud system, the difference in service cost may be several times, dozens, or even dozens of times normally.

Then, we varied the number of jobs from 260 to 460 in steps of 20 on a cloud system with 200 VMs, the simulation results of which are shown in Fig. 6. For the average cost, we can observe from Fig. 6a that our proposed RCJS is better than MIN-MIN for 260, 280, and 300 jobs and the performance of these two algorithms are almost the same when the number of jobs in the range 320 to 400, while for others,

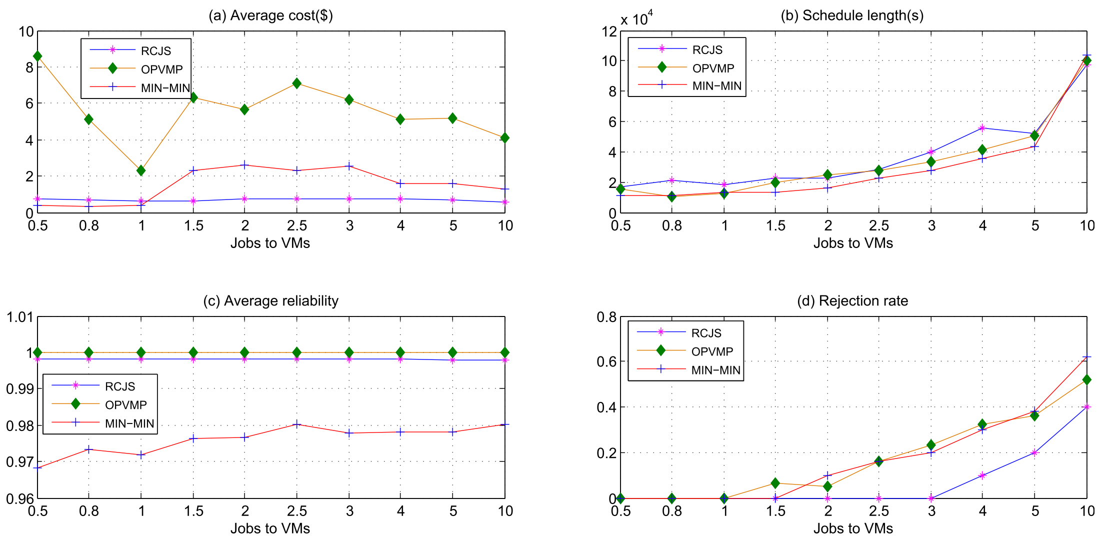


Fig. 7. The comparison results of the ratio of jobs to VMs.

TABLE 4
Some Cloud Services Scheduling Solutions of Three Algorithms for 400 Jobs on 200 VMs

	RCJS					OPVMP					MIN-MIN				
ServiceID	Cost	Reliability	Finish Time	Rejection	VMs	Cost	Reliability	Finish Time	Rejection	VMs	Cost	Reliability	Finish Time	Rejection	VMs
240	1.759	0.954	26986	No	1	2.129	1.0	22244	No	2	0.481	0.9711	8840	No	1
241	0.032	0.998	2549	No	1	3.89	1.0	1522	No	4	0.61	0.984	1141	No	1
242	5.73	1.0	26494	No	2	15.07	1.0	7822	No	4	1.736	0.971	6551	No	1
243	0.406	0.999	10663	No	1	8.83	1.0	8687	No	2	0.698	0.958	7494	No	1
244	0.125	0.982	9738	No	1	7.668	1.0	7465	No	4	1.645	0.91	7388	No	1
245	0.63	0.967	5578	No	1	2.63	1.0	1236	No	4	0.803	0.9990	1201	No	1
246	0.228	0.982	9273	No	1	0.815	1.0	8768	No	2	1.331	0.876	9088	No	1
247	1.293	0.976	2329	No	1	16.4	1.0	2927	No	2	1.34	0.981	2401	No	1
248	0.021	0.985	2117	No	1	1.207	1.0	2582	No	4	0.471	0.95	2547	No	1
249	0.097	0.96	10126	No	1	2.01	1.0	9020	No	2	1.595	0.966	8637	No	1
250	0.689	0.965	3042	No	1	14.99	1.0	3765	No	3	2.693	0.956	3299	No	1
251	6.592	0.936	3559	No	1				Yes					Yes	
252	0.356	0.974	4793	No	1	1.228	1.0	978	No	3	2.176	0.881	948	No	1
253	1.503	0.979	5414	No	1	1.636	1.0	9179	No	2	1.5	0.986	8518	No	1
254	1.502	0.967	5372	No	1	6.918	1.0	7631	No	2	1.423	0.983	7333	No	1
255	1.256	0.973	14293	No	1	2.44	1.0	4290	No	2	3.885	0.958	4188	No	1
256	8.452	0.981	4205	No	1				Yes		1.278	0.971	7401	No	1
257	0.127	0.983	11724	No	1	9.23	1.0	8302	No	2	1.742	0.963	8133	No	1
258	7.555	1.0	27467	No	2	2.55	1.0	6198	No	2	1.714	0.982	4138	No	1
259	0.131	0.985	4118	No	1	4.32	1.0	688	No	4	1.39	0.928	609	No	1

our proposed RCJS is inferior to MIN-MIN. Further, RCJS outperforms OPVMP in terms of average cost by 205.2%. This is primarily because most jobs scheduled by the OPVMP algorithm have multiple backups, which require more computational costs and result in high reliability. This phenomenon can be also concluded from Fig. 6c, where RCJS and MIN-MIN algorithms are inferior to OPVMP by 1.8% and 6.9% in term of average reliability, respectively. In fact, the OPVMP algorithm achieves high reliability by consuming large amounts of computational resources. However, this multi-backup strategy regardless of cost is not the optimal solution. We can achieve an effective tradeoff between multi-backup and reliability.

For the rejection rate (in Fig. 6d), our proposed RCJS algorithm significantly outperforms OPVMP, MIN-MIN by 93.1%, 73.6%, respectively. The rejection rate increases as the number of jobs increases. This is because with certain VMs in cloud systems, more jobs will degrade the system performance. Therefore, our proposed RCJS is superior to OPVMP in terms of average cost, rejection rate, and outperforms MIN-MIN in terms of average reliability, rejection rate. The MIN-MIN algorithm has the best performance in

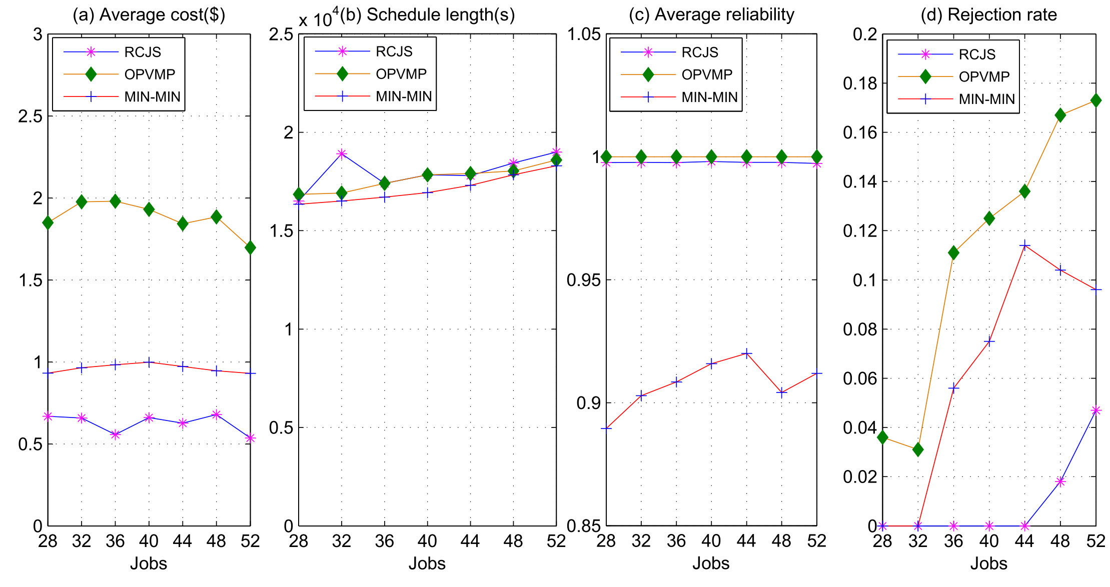


Fig. 8. The comparison results of real-world platform.

schedule length (in Fig. 6b) among all three algorithms. This is a reasonable performance, although the MIN-MIN algorithm optimizes the cloud service processing time regardless of computation cost, reliability, and rejection rate.

Fig. 7 shows the simulation results of the ratio of jobs to VMs on cloud systems, where we varied this ratio within [0.5, 10]. By using a range of the ratio of jobs to VMs, different cloud systems workload can be accommodated. That is, systems with low workload may be simulated by assuming the ratio as 0.5, whereas, the high workload system is set as 10. From Fig. 7, we can also conclude that our proposed RCJS algorithm demonstrates a good trade-off among the factors of average cost, average reliability, and schedule length, when compared to the other algorithms. This algorithm also demonstrates superior performance over OPVMP, MIN-MIN in term of rejection rate.

6.4 Performance Results of Real-World Platform

We also conducted the performance evaluation on the real-world experimental environment. The cloud service jobs varied from 28 to 52 in steps of 4. The experimental results are shown in Fig. 8. We can observe from Fig. 8a that our proposed RCJS algorithm demonstrates better performance than OPVMP by 66.7% and MIN-MIN by 34.8% for the parameter of average cost. For the reliability in Fig. 8c, both the RCJS and OPVMP algorithms demonstrate high reliability; moreover, both algorithms are obviously better than the MIN-MIN algorithm. This performance improvement can also be observed in Fig. 8d, where RCJS outperforms OPVMP by 91.6% and MIN-MIN by 85.3% in terms of the rejection rate. For the schedule length, the optimal scheduling algorithm is the traditional MIN-MIN algorithm. Therefore, our proposed RCJS strategy demonstrates good cloud computing system performance trade-off and is suitable for cloud services with high reliability and low-cost requirements.

7 Conclusions and Future Work

This paper investigates the problem of scheduling cloud services on VMs that are under a response time constraint. In order to satisfy the high reliability and low-cost requirements of cloud services, we proposed a reliability and cost aware job scheduling scheduling (RCJS) strategy on cloud computing systems. We initially implemented a resources management framework and analyzed the cloud service reliability. Then, we formulated this cloud service scheduling as a linear programming problem that attempts to minimize the total cost and rejection rate. Finally, we proposed the RCJS algorithm, which uses a modified cloud service cost based on the response time slack factor α.

The experimental results with 200 VMs demonstrated that the RCJS algorithm significantly outperforms OPVMP in term of average cost by 205.2%, and RCJS is also better than OPVMP, MIN-MIN by 93.1%, 73.6%, in term of rejection rate. For real-world platform, our proposed RCJS also has good performance in terms of average service cost and rejection rate, and get good trade-off in reliability when compared to the other algorithms.

Future studies on reliability and cost aware cloud services scheduling approach can be conducted in the following four directions. First, the reliability of the cloud system will decrease with the increase of the energy consumption. Therefore, it will be interesting work to consider the energy consumption of cloud systems. Second, the effectiveness of the systems reliability analysis and cloud service cost calculation demonstrate significant room for improvement. Third, the rejected jobs can be rescheduled if the service's response time is satisfied. Here, we can remove some jobs with long response slack time from its VMs. Therefore, the cloud systems has the room to accommodate rejected jobs. In the rescheduling stage, we can assign higher rescheduling priority to these rejected jobs to reduce the rejection rate. Fourth, employing other platforms such as Google Cloud or Microsoft Azure is a very interesting comparison work, we plan to extend it in our future work.

References

[1] F. Fowley, C. Pahl, P. Jamshidi, D. Fang, and X. Liu, "A classification and comparison framework for cloud service brokerage architectures," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 358–371, Apr.–Jun. 2018.

[2] L. Sun, J. Ma, H. Wang, Y. Zhang, and J. Yong, "Cloud service description model: An extension of USDL for cloud services," *IEEE Trans. Services Comput.*, vol. 11, no. 2, pp. 354–368, Mar./Apr. 2018.

[3] G. L. Stavrinides and H. D. Karatza, "An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations," *Future Gener. Comput. Syst.*, vol. 96, pp. 216–226, Jul. 2019.

[4] K. Gu, N. Wu, B. Yin, and W. Jia, "Secure data query framework for cloud and fog computing," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 332–345, Mar. 2020.

[5] A. Zhou *et al.*, "Cloud service reliability enhancement via virtual machine placement optimization," *IEEE Trans. Services Comput.*, vol. 10, no. 6, pp. 902–913, Nov./Dec. 2017.

[6] X. Li, Y. Liu, R. Kang, and L. Xiao, "Service reliability modeling and evaluation of active-active cloud data center based on the IT infrastructure," *Microelectron. Rel.*, vol. 75, pp. 271–282, Aug. 2017.

[7] Y. Sharma, W. Si, D. Sun, and B. Javadi, "Failure-aware energy-efficient VM consolidation in cloud computing systems," *Future Gener. Comput. Syst.*, vol. 94, pp. 620–633, May 2019.

[8] P. Institute, "Cost of data center outages." Jan. 2016, Accessed: Aug. 10, 2019. [Online]. Available: https://planetaklimata. com. ua/instr/Liebert_Hiross/Cost_of_Data_Center_Outages_201 6_Eng.pdf

[9] K. Vinay, S. M. Dilip Kumar, S. Raghavendra, and K. R. Venugopal, "Cost and fault-tolerant aware resource management for scientific workflows using hybrid instances on clouds," *Multimedia Tools Appl.*, vol. 77, no. 8, pp. 10 171–10 193, Apr. 2018.

[10] F. Machida, M. Kawato, and Y. Maeno, "Redundant virtual machine placement for fault-tolerant consolidated server clusters," in *Proc. IEEE Netw. Oper. Manage. Symp.*, 2010, pp. 32–39.

[11] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue, "Survivable virtual infrastructure mapping in virtualized data centers," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, 2012, pp. 196–203.

[12] S. Homsi, S. Liu, G. A. Chaparro-Baquero, O. Bai, S. Ren, and G. Quan, "Workload consolidation for cloud data centers with guaranteed QoS using request reneging," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 2103–2116, Jul. 2017.

[13] G. Xie, G. Zeng, R. Li, and K. Li, "Quantitative fault-tolerance for reliable workflows on heterogeneous IaaS clouds," *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 1223–1236, Oct.–Dec. 2020.

[14] H. Xu, X. Qiu, Y. Sheng, L. Luo, and Y. Xiang, "A QoS-Driven approach to the cloud service addressing attributes of security," *IEEE Access*, vol. 6, pp. 34 477–34 487, Jun. 2018.

[15] P. Sigdel, X. Yuan, and N.-F. Tzeng, "Realizing best checkpointing control in computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 315–329, Feb. 2021.

[16] X. Zhu, J. Wang, H. Guo, D. Zhu, L. T. Yang, and L. Liu, "Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3501–3517, Dec. 2016

[17] A. Dogan and F. Zgner, "Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 308–323, Mar. 2002.

[18] X. Tang, K. Li, and G. Liao, "An effective reliability-driven technique of allocating tasks on heterogeneous cluster systems," *Cluster Comput.*, vol. 17, no. 4, pp. 1413–1425, Dec. 2014.

[19] H. Balla, G. Chen, and W. Jing, "Reliability enhancement in cloud computing via optimized job scheduling implementing reinforcement learning algorithm and queuing theory," in *Proc. 1st Int. Conf. Data Intell. Secur.*, 2018, pp. 127–130.

[20] M. Jawad *et al.*, "A robust optimization technique for energy cost minimization of cloud data centers," *IEEE Trans. Cloud Comput.*, vol. 9, no. 2, pp. 447–460, Apr.–Jun. 2021.

[21] Z. Ye, S. Mistry, A. Bouguettaya, and H. Dong, "Long-term QoS-aware cloud service composition using multivariate time series analysis," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 382–393, May/Jun. 2016.

[22] X. Tang, "Large-scale computing systems workload prediction using parallel improved LSTM neural network," *IEEE Access*, vol. 7, pp. 40 525–40 533, Mar. 2019.

[23] A. Rayan and Y. Nah, "Energy-aware resource prediction in virtualized data centers: A machine learning approach," in *Proc. IEEE Int. Conf. Consum. Electron.*, 2018, pp. 206–212.

[24] Y. Lu *et al.*, "Service deployment and scheduling for improving performance of composite cloud services," *Comput. Elect. Eng.*, vol. 74, pp. 616–634, Mar. 2019.

[25] Y. Lian, Y. Tang, and Y. Wang, "Objective Bayesian analysis of JM model in software reliability," *Comput. Statist. Data Anal.*, vol. 109, pp. 199–214, May 2017.

[26] Z. Wen, J. Caa, P. Watson, and A. Romanovsky, "Cost effective, reliable and secure workflow deployment over federated clouds," *IEEE Trans. Services Comput.*, vol. 10, no. 6, pp. 929–941, Nov./Dec. 2017.

[27] A. Ouyang, Z. Tang, X. Zhou, Y. Xu, and K. Li, "Parallel hybrid PSO with CUDA for lD heat conduction equation," *Comput. Fluids*, vol. 110, pp. 198–210, Mar. 2015.

[28] T. Braun *et al.*, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *J. Parallel Distrib. Comput.*, vol. 61, no. 6, pp. 810–837, Jun. 2001.

[29] D. Kondo, B. Javadi, A. Iosup, and D. Epema, "The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems," in *Proc. 10th Int. Conf. Cluster Cloud Grid Comput.*, 2010, pp. 398–407.

[30] J. Wang and C. Zhang, "Software reliability prediction using a deep learning model based on the RNN encoder-decoder," *Rel. Eng. Syst. Saf.*, vol. 170, pp. 73–82, Feb. 2018.

[31] A. Amin, L. Grunske, and A. Colman, "An approach to software reliability prediction based on time series modeling," *J. Syst. Softw.*, vol. 86, no. 7, pp. 1923–1932, Jul. 2013.

[32] P. K. Sahoo, C. K. Dehury, and B. Veeravalli, "LVRM: On the design of efficient link based virtual resource management algorithm for cloud platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 4, pp. 887–900, Apr. 2018.

[33] H. Singh, S. Tyagi, P. Kumar, S. S. Gill, and R. Buyya, "Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions," *Simul. Modelling Pract. Theory*, vol. 111, Sep. 2021, Art. no. 102353.

[34] Amazon EC2. Jul. 2018, Accessed: Apr. 25, 2021. [Online]. Available: http://aws.amazon.com/ec2/

[35] W. Li, P. Zhang, H. Leung, and S. Ji, "A novel QoS prediction approach for cloud services using Bayesian network model," *IEEE Access*, vol. 6, pp. 1391–1406, Feb. 2018.

[36] M. Eisa, M. Younas, K. Basua, and I. Awan, "Modelling and simulation of QoS-aware service selection in cloud computing," *Simul. Modelling Pract. Theory*, vol. 103, Sep. 2020, Art. no. 102108.

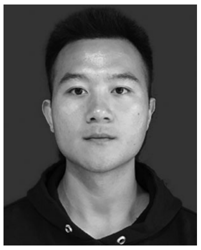
[37] S. Long, W. Long, Z. Li, K. Li, Y. Xia, and Z. Tang, "A game-based approach for cost-aware task assignment with QoS constraint in collaborative edge and cloud environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1629–1640, Jul. 2021.

[38] P. Nawrocki and P. Osypanka, "Cloud resource demand prediction using machine learning in the context of QoS parameter," *J. Grid Comput.*, vol. 19, May 2021, Art. no. 20.

[39] B. Sniezynski, P. Nawrocki, M. Wilk, M. Jarzab, and K. Zielinski, "VM reservation plan adaptation using machine learning in cloud computing," *J. Grid Comput.*, vol. 17, pp. 797–812, Jul. 2019.

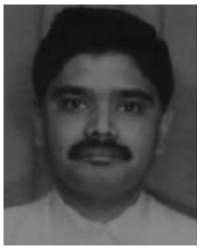
[40] X. Chen, H. Wang, X. Zheng, and L. Guo, "Self-adaptive resource allocation for cloud-based software services based on iterative QoS prediction model," *Future Gener. Comput. Syst.*, vol. 105, pp. 287–296, Apr. 2020.

Xiaoyong Tang received the MS and PhD degrees from Hunan University, Changsha, China, in 2007 and 2013, respectively. He is currently a professor with the School of Computer and Communication Engineering, Changsha University of Science and Technology. His research interests include parallel computing, cloud computing, big data, modeling and scheduling of distributed computing systems, distributed system reliability, and parallel algorithms. He has published more than 50 technique papers in international journals and conferences. He holds more than 10 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the World's Top 2% Scientists. He is the reviewers of *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Industrial Informatics*, *Journal of Parallel and Distributed Computing*, *Future Generation Computer Systems*, CPE, JS, and so on.



Yi Liu is currently working toward the master's degree in the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, China, since 2020. His research interests include deep learning, parallel distributed computing, and embedded development. He has one patent and one software copyright announced or authorized by the Chinese National Intellectual Property Administration.

Zeng Zeng (Senior Member, IEEE) received the PhD degree in electrical and computer engineering from the National University of Singapore, Singapore. Currently, he works as a senior scientist, program head, I2R, A*Star, Singapore. From 2011 to 2014, he worked as a senior research fellow with the National University of Singapore. From 2005 to 2011, he worked as a professor with the Computer and Communication School, Hunan University, China. His research interests include distributed/parallel computing systems, data stream analysis, deep learning, multimedia storage systems, wireless sensor networks.



Bharadwaj Veeravalli (Senior Member, IEEE) received the BSc degree in physics from Madurai-Kamaraj Uiversity, Tamil Nadu, India, in 1987, the master's degree in electrical communication engineering from the Indian Institute of Science, Bangalore, India, in 1991, and the PhD degree from the Department of Aerospace Engineering, Indian Institute of Science, Bangalore, India, in 1994. He did his postdoctoral research with the Department of Computer Science, Concordia University, Montreal, Canada, in 1996. He is currently with the Department of Electrical and Computer Engineering, National University of Singapore, as a tenured associate professor. His research interests include multiprocessor systems, cluster/grid computing, scheduling in parallel and distributed systems, bioinformatics & computational biology, and multimedia computing. He is one of the earliest researchers in the field of divisible load theory (DLT). He is currently serving the editorial board of the *IEEE Transactions on Computers*, *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, and *International Journal of Computers & Applications*, USA, as an associate editor.