Using Proactive Fault-Tolerance Approach to Enhance Cloud Service Reliability

Jialei Liu, Shangguang Wang, *Senior Member, IEEE*, Ao Zhou, Sathish A. P. Kumar, *Senior Member, IEEE*, Fangchun Yang, and Rajkumar Buyya, *Fellow, IEEE*

Abstract—The large-scale utilization of cloud computing services for hosting industrial/enterprise applications has led to the emergence of cloud service reliability as an important issue for both cloud service providers and users. To enhance cloud service reliability, two types of fault tolerance schemes, reactive and proactive, have been proposed. Existing schemes rarely consider the problem of coordination among multiple virtual machines (VMs) that jointly complete a parallel application. Without VM coordination, the parallel application execution results will be incorrect. To overcome this problem, we first propose an initial virtual cluster allocation algorithm according to the VM characteristics to reduce the total network resource consumption and total energy consumption in the data center. Then, we model CPU temperature to anticipate a deteriorating physical machine (PM). We migrate VMs from a detected deteriorating PM to some optimal PMs. Finally, the selection of the optimal target PMs is modeled as an optimization problem that is solved using an improved particle swarm optimization algorithm. We evaluate our approach against five related approaches in terms of the overall transmission overhead, overall network resource consumption, and total execution time while executing a set of parallel applications. Experimental results demonstrate the efficiency and effectiveness of our approach.

Index Terms—Cloud data center, cloud service reliability, fault tolerance (FT), particle swarm optimization (PSO), virtual cluster

1 Introduction

Cloud computing is widely adopted in current professional and personal environments. It employs several existing technologies and concepts, such as virtual servers and data centers, and gives them a new perspective [1]. Furthermore, it enables users and businesses to not only use applications without installing them on their machines but also access resources on any computer via the Internet [2]. With its pay-per-use business model for customers, cloud computing shifts the capital investment risk for under- or overprovisioning to cloud providers. Therefore, several leading technology companies, such as Google, Amazon, IBM, and Microsoft, operate large-scale cloud data centers around the world. With the growing popularity of cloud computing, modern cloud data centers are employing tens of thousands of physical machines (PMs) networked via hundreds of routers/switches that communicate and coordinate to deliver highly reliable cloud computing services. Although the failure probability of a single device/link might be low [3], it is magnified across all the devices/links hosted in a cloud data center owing to the problem of coordination of PMs. Moreover, multiple fault sources (e.g., software, human errors, and hardware) are the norm rather than the exception [4]. Thus, downtime is common and seriously affects the service level of cloud computing [5]. Therefore, enhancing cloud service reliability is a critical issue that requires immediate attention.

Over the past few years, numerous fault tolerance (FT) approaches have been proposed to enhance cloud service reliability [6], [7]. It is well known that FT consists of fault detection, backup, and failure recovery, and nearly all FT approaches are based on the use of redundancy. Currently, two basic mechanisms, namely, replication and checkpointing, are widely adopted. In the replication mechanism, the same task is synchronously or asynchronously handled on several virtual machines (VMs) [8], [9], [10]. This mechanism ensures that at least one replica is able to complete the task on time. Nevertheless, because of its high implementation cost, the replication mechanism is more suitable for real time or critical cloud services. The checkpointing mechanism is categorized into two main types: independent checkpoint mechanisms that only consider a whole application to perform on a VM, and coordinated checkpoint mechanisms that consider multiple VMs (i.e., a virtual cluster) to jointly execute parallel applications [11], [12], [13], [14], [15], [16]. The two types of mechanisms periodically save the execution state of a running task as a checkpoint image file. When downtime occurs, they can resume the task on a different PM based on the last saved checkpoint image. In other words, the task need not be restarted from the

- *J. Liu is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China and the Department of Computer Science and Information Engineering, Anyang Institute of Technology, Anyang, China. E-mail: liujialei@bupt.edu.cn.*
- *S. Wang, A. Zhou, and F. Yang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. E-mail: {sgwang, aozhou, fcyang}@bupt.edu.cn.*
- *S.A.P. Kumar is with the Department of Computer Science and Information Systems, Coastal Carolina University, Conway, SC 29528-6054. E-mail: skumar@coastal.edu.*
- *R. Buyya is with Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, The University of Melbourne, Melbourne, Vic. 3010, Australia. E-mail: rbuyya@unimelb.edu.au.*

Manuscript received 20 Sept. 2015; revised 15 Mar. 2016; accepted 29 Apr. 2016. Date of publication 0 . 2016
Recommended for acceptance by D. Lie.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2016.2567392

beginning but only from the last saved state. Thus, checkpointing can reduce lost time due to PM faults and improve cloud service reliability.

Existing FT approaches can also be classified into two other types: reactive schemes, which lead to temporal service downtime or performance degradation, and proactive schemes based on the failure prediction of the PM for the Xen virtualization platform [17], [18], [19]. It is well known that current FT techniques focus on reactive schemes to recover from faults and generally rely on a checkpoint/restart mechanism. However, when the application behavior is highly dynamic (e.g., social networks), reactive schemes can produce poor performance and may lead to low average utilization of resources. In current systems, PM failures can often be anticipated on the basis of deteriorating health status by monitoring fan speed, CPU temperature, memory, and disk error logs. Therefore, instead of a reactive FT scheme, a proactive scheme that adopts a PM monitoring scheme to detect a deteriorating PM can be used [18], [20]. This approach can reduce checkpoint frequencies as fewer unanticipated failures are encountered, and it is complementary to reactive FT. Reactive and proactive schemes often consider a whole parallel application to execute on a VM. However, they rarely consider a virtual cluster, which consists of multiple VMs distributed across PMs, to collectively execute distributed applications (e.g., client-server systems, parallel programs, and transaction processing). Unfortunately, the failure of a single VM usually causes a significant crash or fault in other related parts of the virtual cluster. Therefore, it is important to deal with this situation effectively and efficiently.

It is well known that a virtual cluster works in a cooperative manner to process parallel applications, and intermediate results are transferred among them iteratively through multiple stages. Moreover, the traffic generated by these applications creates flows not only between VMs but also into the Internet [21]. It often contributes a significant portion of the running time of a parallel application, e.g., 26 percent of the jobs in a Facebook data center spend more than 50 percent of their running times in transferring data [22]. With the growing number of parallel applications required to process big data in cloud data centers, cloud data center traffic is increasing rapidly. Recently, Cisco predicted that the global cloud data center traffic will nearly triple from 2013 to 2018 with a combined annual growth rate of 23 percent, i.e., from 3.1 ZB/year in 2013 to 8.6 ZB/year in 2018 [23]. Therefore, in cloud data centers shared by many parallel applications, the upper-level bandwidth resources, especially the bandwidth resources of the core layer, of the cloud data center network may become a bottleneck [24]. Furthermore, interference due to parallel application traffic in the network could result in unpredictable running times, which could adversely affect cloud service reliability and lead to financial losses.

To overcome the upper-level bandwidth resource bottlenecks and enhance cloud service reliability, this paper proposes a proactive coordinated FT (PCFT) approach based on particle swarm optimization (PSO) [25], which addresses the proactive coordinated FT problem of a virtual cluster with the objective of minimizing the overall transmission overhead, overall network resource consumption, and total execution time while executing a set of parallel applications.

The **key contributions** of our work can be summarized as follows.

- First, we introduce a deteriorating PM modeling problem, and then we propose a coordinated FT problem of the VMs on the detected deteriorating PM to search for some optimal target PMs for these VMs.
- To solve the two above-mentioned problems, we propose the PCFT approach, which is realized in two steps: first, we introduce a PM fault prediction model to proactively anticipate a deteriorating PM, and then, we improve the PSO algorithm to solve the coordinated FT problem.
- We set up a system model to evaluate the efficiency and effectiveness of the proposed PSO-based PCFT approach by comparing it with five other related approaches in terms of overall transmission overhead, overall network resource consumption, and total execution time while executing a set of parallel applications.

The remainder of this paper is organized as follows. Section 2 introduces the background and related work. Section 3 describes the VM coordinated mechanism and the system model of the proposed approach. Section 4 provides the technical details of the proposed approach. Section 5 discusses the performance evaluation, including the experimental parameter configuration, comparison results, and effects of the experimental parameters. Finally, Section 6 concludes this paper with recommendations for future work.

2 Background and Related Work

To enhance cloud service reliability, numerous FT approaches have been proposed, which adopt the redundant VM placement approach for multiple applications [17], [26], [27]. The main concept underlying these approaches is to ensure that all cloud services can be maintained while any k PMs fail at the same time. Remus is a practical high-availability service that enables a running system to transparently continue execution on an alternate PM in the event of failure with only a few seconds of downtime [17]. However, Remus only provides an asynchronous VM replication mechanism for an individual VM. Deng et al. proposed a novel offloading system to design robust offloading decisions for mobile cloud services. They design a trade-off FT mechanism for the offloading system, which not only reinitiates lost commutating tasks, but also minimizes the extra execution time and energy consumption caused by failures [28].

Moreover, in the cloud computing environment, in addition to ensuring cloud service reliability, cloud service FT approaches should reduce resource consumption as much as possible on the basis of the cloud data center characteristics, for example, Wang et al. proposed a VM placement method in national cloud data centers for the first time, which provide a good solution for operating green and reliable national cloud data centers [29]. Because of the high costs incurred by the replication mechanism, approaches based on it are suitable only for critical tasks. To overcome this problem, notable approaches have been introduced to identify the significant parts of a complex task in order to reduce the implementation cost [9], [10], [30], [31]. These

notable approaches first calculate the significance value of each subtask according to the invocation structures and frequencies [9], [10], [30]. Then, they rank the subtasks on the basis of the calculated significance values and determine the redundancy of each subtask accordingly. Unlike the fixed redundancy level approach, these approaches can reduce the implementation cost by changing the redundancy of a component when a failure occurs [8]. In spite of the above-mentioned improvement, the implementation of the replication mechanism remains a costly task. Thus, such a mechanism is more suitable for real time or critical tasks.

Nevertheless, for some non-real-time large-scale tasks, a widely used FT technique called checkpointing is relatively more effective [32], [33]. In general, checkpointing is categorized into the independent checkpoint mechanism [12], [13], [18], [34] and the coordinated checkpoint mechanism [16], [35], [36]. From the viewpoint of independent checkpointing, Nagarajan et al. proposed a proactive FT mechanism that can anticipate a deteriorating PM through resource monitoring, i.e., monitoring CPU temperature, memory, fan speed, and disk logs, and migrate VMs on the deteriorating PM to healthy PMs before the occurrence of any failure [18]. Recently, Liu et al. made a pioneering effort to proactively measure and improve the imperfect cache mechanism of current mobile cloud computing applications [37]. Overall, the proactive FT mechanism is complementary to reactive FT using full checkpoint schemes, because the checkpoint frequencies can be reduced as fewer unanticipated failures are encountered. Although a virtual cluster is considered to collectively execute parallel applications, the migration technique is adopted to enhance reliability. Dynamic checkpointing strategies developed by investigating and analyzing the independent checkpoint mechanism can significantly reduce costs while improving reliability [34]. Goiri et al. presented a smart checkpoint infrastructure that uses Another Union File System to differentiate read-only parts from read-write parts of the virtual machine image for virtualized service providers [12]. Although this approach is an effective way to resume a task execution faster after a node crash and to increase the FT of the system, it overlooks the fact that the core switches are the bottleneck of the cloud data center network. When the checkpoint images are stored in central storage PMs, the checkpoint traffic may congest the core switches, which affects the FT. To overcome this problem, Zhou et al. proposed a cloud service reliability enhancement approach for minimizing network and storage resource usage in a cloud data center [13]. In their approach, the identical parts of all VMs that provide the same service are checkpointed once as the service checkpoint image. Moreover, the remaining checkpoint images only save the modified page. This approach not only guarantees cloud service reliability but also consumes less network and storage resources than other approaches.

Although several checkpoint mechanisms have been introduced, as discussed above, they rarely consider the consistency of virtual clusters. To deal with this situation, the coordinated checkpoint mechanism has been proposed [16], [35], [36]. In order to minimize the performance loss due to unexpected failures or unnecessary overhead of FT mechanisms, Liu et al. proposed an optimal coordinated checkpoint placement scheme to cope with different failure distributions and a varying checkpoint interval [35]. This scheme also considers optimality for both checkpoint overhead and rollback time. Zhang et al. proposed VirtCFT, a system-level coordinated distributed checkpointing FT system that provides FT for a virtual cluster and recovers the entire virtual cluster to the previous correct state when a fault occurs by transparently taking incremental checkpoints of VM images [16]. Considering that users' individual requirements may vary considerably, Limrungsi et al. proposed a novel scheme for providing reliability as a flexible on-demand service [36]. This scheme uses peer-to-peer checkpointing and allows user reliability levels to be jointly optimized by assessing users' individual requirements and total available resources in the cloud data center.

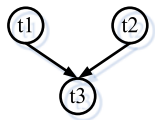


Fig. 1. Parallel application model.

Although the proactive FT scheme and virtual clusters have been widely adopted [21], [22], [24], they are rarely used together to enhance the reliability of cloud data centers. Therefore, this paper proposes a CPU temperature model for anticipating a deteriorating PM. In order to reallocate the VMs on the detected deteriorating PM as compactly as possible to other VMs in the same virtual cluster, the PSO-based PCFT approach is introduced to identify some optimal PMs for these VMs.

3 Preliminaries and System Model

In order to make it easier to understand our approach, we first introduce the basic knowledge of the VM coordinated mechanism and then propose our system model.

3.1 VM Coordinated Mechanism

In this section, a VM coordinated mechanism (i.e., virtual cluster) is designed to jointly process a set of parallel applications (e.g., web applications), and each parallel application includes multiple tasks. However, for ease of understanding, a parallel application model (see Fig. 1) [38], [39], which is considered to be a data-intensive application, is proposed as our test case to measure the performance of different approaches in terms of the overall network resource consumption and total execution time. Each parallel application consists of three tasks (t1, t2, and t3); t3 cannot enter the execution stage until both t1 and t2 transfer data to t3. Each task, which is executed by a VM, consists of some computation and communication stages.

3.2 System Model

In this paper, the target system is an IaaS environment that employs a fat-tree topology architecture (see Fig. 2) [40]. The advantage of using this topology is that all switches are identical commodity Ethernet switches. Moreover, this topology has the potential to deliver large bisection bandwidth through rich path multiplicity for relieving the bandwidth resource bottlenecks.

In the fat-tree topology architecture, there are n heterogeneous PMs, which have different resource capacities, and

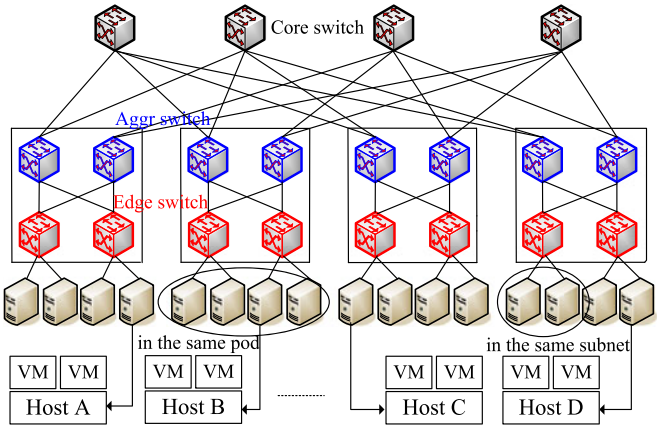


Fig. 2. Fat-tree topology architecture (the switches in the top (black), middle (blue), and bottom (red) layers are the core, aggregation, and edge switches, respectively).

three-level trees of switches. Each PM is characterized by the CPU performance defined in millions of instructions per second (MIPS), amount of RAM, network bandwidth, and disk storage. At any given time, a cloud data center usually serves many simultaneous users. Users submit their requests for provisioning n heterogeneous VMs, which are allocated to the PMs and characterized by requirements of CPU performance, RAM, network bandwidth, and disk storage. The length of each request is specified in millions of instructions. The bottom layer is the edge layer; the switches in this layer are edge switches that can attach to the network. The link that connects an edge switch and a PM is an edge link. All PMs physically connected to the same edge switch form their own subnet. The middle layer is the aggregation layer, and its switches are aggregation switches. The link that connects a core switch and an aggregation switch is an aggregation link. All PMs that share the same aggregation switches are in the same pod. The top layer is the core tier, and the switches in this layer are core switches. The link that connects a core switch and an aggregation switch is a core link. Because all traffic moving outside the cloud data center should be routed through the core switch, the core link becomes congested easily. Consequently, we should try to reduce the network resource consumption of the core link.

In our PCFT approach, multiple VMs (i.e., a virtual cluster) jointly complete a set of parallel applications. We choose three VMs as a virtual cluster when creating the VMs. To initially allocate these VMs to the PMs, we design the Initial Virtual Cluster Allocation algorithm (IVCA), which reduces the resource consumption as much as possible. The pseudocode of the IVCA approach is presented in Algorithm 1. When a VM is allocated to a PM, IVCA first traverses all the PMs in the cloud data center to identify all other VMs that are in the same virtual cluster as the VM. If such VMs exist, the VM is allocated to the same subnet or pod as the PM hosting such VMs. Otherwise, it will be allocated such that the total energy consumption of all PMs in the target system is minimized. Thus, each VM is allocated to a PM that provides the least increase in the network resource consumption and energy consumption of all the PMs in the target system.

In general, tens of thousands of PMs and a multitenancy model are employed in a production environment. Therefore, downtime is common and seriously affects the service

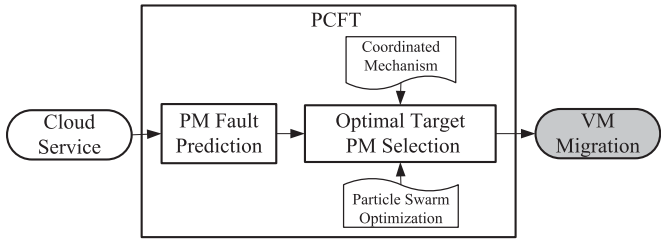


Fig. 3. PCFT architecture.

level of cloud computing. Therefore, we focus on PM fault features to anticipate a deteriorating PM. The deteriorating PM is selected on the basis of the CPU temperature model, which is introduced in Section 4.1. This model is used to determine when the temperature exceeds the upper threshold of the normal CPU temperature range (e.g., 68 °C) for the duration in which the PM is considered to be deteriorating. Then, the VM reallocation algorithm is adopted to reallocate the VMs on the deteriorating PM to other healthy PMs.

Algorithm 1: Initial Virtual Cluster Allocation (IVCA)

```
1: Input: hostList, vmList Output: allocation scheme of VMs
2: foreach vm in vmList do
3:     minPower←MAX
4:     foreach host in hostList do
5:         foreach vm1 of vmList in the host do
6:             if vm1 and vm are in the same virtual cluster then
7:                 allocate vm1 to the same subnet or pod as the host
8:     foreach host in hostList do
9:         if host has sufficient resources for vm then
10:            power←energyFitness(globalBestList, hostList)
11:            if power< minPower then
12:                targetHost←host
13:                minPower←power
14:        if targetHost≠NULL then
15:            allocate vm to targetHost
16: return allocation scheme of VMs
```

4 Proposed PCFT Approach

The health monitoring mechanism is adopted to guarantee cloud service reliability in our approach (PCFT). The objective of the PCFT approach is to monitor and anticipate a deteriorating PM. When there exists a deteriorating PM, our approach will search for some optimal target PMs for the VMs hosted on the deteriorating PM.

As shown in Fig. 3, the system architecture of our approach consists of the following two modules.

- PM fault prediction: CPU temperature monitoring and forecasting are essential for preventing PM shutdowns due to overheating as well as for improving the data center's energy efficiency. The module has a prediction functionality to monitor and anticipate a deteriorating PM by limiting the CPU temperature in the normal temperature range.
- Optimal target PM selection: When the deteriorating PM is detected, the module searches for optimal target PMs for the VMs on the deteriorating PM. To search for these optimal target PMs and to execute a cloud service that consists of a set of parallel

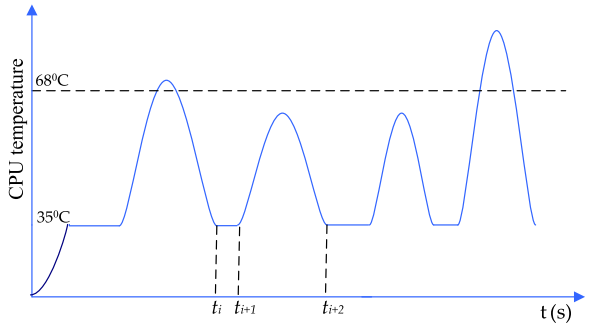


Fig. 4. Corresponding curve of (1).

applications, we design a VM coordinated mechanism by selecting three VMs as a virtual cluster to jointly execute a parallel application and model the optimal target PM selection as a PSO-based optimization problem within constraints.

In the following sections, we will describe the details of PM fault prediction, optimal target PM selection, and PSO-based PM selection optimization.

4.1 PM Fault Prediction

Thermal performance is a critical metric in cloud data center management, and sharp spikes in PM utilization may result in disruptive downtimes due to generated hotspots. CPU temperature is an approximately linear function of CPU utilization without considering the effect of heat generated by other PMs nearby [41]. Thus, we focus on PM fault features to anticipate a deteriorating PM, and the deteriorating PM is selected on the basis of CPU temperature. As the fault metrics are extensible, we plan to study additional metrics (fan speed and voltage, disk error logs, etc.) in the future.

CPU temperature monitoring and forecasting are essential for preventing PM shutdowns due to overheating as well as for improving the data center's energy efficiency. Therefore, the simulated prediction function model for CPU temperature in the data center is modeled as follows, and the corresponding curve is shown in Fig. 4 [42], [43]:

$$f(t|A,\omega,t_i,t_{i+1}) = \begin{cases} e^t & 0 \le t \le t_i \\ e^{t_i} & t_i \le t \le t_{i+1} \\ A\sin(\omega t - \omega t_{i+1}) + e^{t_i} & t_{i+1} \le t \le t_{i+2}, \end{cases} \quad (1)$$

where i is the set of positive integers; the first subequatione^tsimulates the process of CPU temperature change during computer boot; t_i is a fixed value computed by $e^{t_i} = 35$; e^{t_i} is the CPU no-load temperature, which is always set at 35°C; t_{i+1} is a random value; t_{i+2} is computed by $t_{i+2} = \pi/\omega + t_{i+1}$; A is the amplitude, which denotes the peak maximum value of CPU temperature (usually lower than 68°C); and ω denotes the duration for which the CPU executes the load. We can randomly adjust the value of A and ω to denote different CPU utilizations in different time domains. We just need the first half cycle of the sinusoidal function; its value can be computed by π/ω.

4.2 Optimal Target PM Selection

4.2.1 Overall Transmission Overhead Model

In this paper, we mainly consider that VMs in a virtual cluster coordinate jointly to execute a parallel application (as shown in Fig. 1). The VMs in the same virtual cluster communicate with each other. The network resource consumption and execution time of a virtual cluster are directly related to the transmission overhead between one VM and other VMs in the same virtual cluster. This is because the lower the transmission overhead, the lower is the communication overhead (e.g., communication time and network resource consumption) when one VM communicates with other VMs in the same virtual cluster. Thus, the overall transmission overhead between VMs on the deteriorating PM, which have been migrated to new PMs, and other VMs in the same virtual cluster is modeled as follows:

$$totalTransOverhead = \sum_{i=1}^{m}\sum_{k=1}^{V} y_{ik} * (bw_{ki}+bw_{ik}), \quad (2)$$

where m is the number of VMs on a deteriorating PM; V is the number of VMs in a virtual cluster; bw_{ik} is the bandwidth value from the ith VM on the deteriorating PM to the kth VM in the same virtual cluster as the ith VM; *and* bw_{ki} is the bandwidth value from the kth VM to the ith VM. Note that if the ith VM (or the kth VM) is a data sender, the value of bw_{ik} (or bw_{ki}) is assigned randomly [44], [45] in a certain range (e.g., [0, 500] MB/s); otherwise, its value is 0. Further, y_{ik} is the transmission overhead between the ith VM migrated to a new PM and other VMs in the same virtual cluster as the ith VM.

4.2.2 Optimal Target PM Selection Model

In this section, we describe how to select the optimal target PMs for the VMs on the deteriorating PM. The optimization objective of the optimal target PM selection problem is to minimize the overall transmission overhead while satisfying the resource requirements. Hence, the overall transmission overhead can be modeled as follows:

$$min \sum_{i=1}^{m}\sum_{k=1}^{V} y_{ik} * (bw_{ki}+bw_{ik}). \quad (3)$$

Such that

$$\sum_{j=1}^{n} x_{ij} = 1, x_{ij} = 0 \text{ or } 1, \quad (4)$$

$$\sum_{i=1}^{M} r_i^{mem} x_{ij} < c_j^{mem} \cap \sum_{i=1}^{M} r_i^{cpu} x_{ij} < c_j^{cpu} \cap \sum_{i=1}^{M} r_i^{bw} x_{ij} < c_j^{bw}, \quad (5)$$

$$\sum_j Flow_{ij} - \sum_l Flow_{li} = \begin{cases} 1 & if\ PM_i\ is\ the\ deteriorating\ PM \\ 0 & otherwise \\ -1 & if\ PM_i\ is\ the\ candidate\ target\ PM, \end{cases} \quad (6)$$

where n is the number of PMs in the cloud data center and M is the number of VMs in the cloud data center. Equation (5) shows that a VM can only be placed on one PM such that $x_{ij} = 1$ if the ith VM is run on the jth PM; otherwise, $x_{ij} = 0$. Equation (6) shows that the sum of the resource requirements of the VMs must be less than the PM's idle resource capacity. Further r_i^{bw}, r_i^{mem}, and r_i^{cpu} are the maximum network bandwidth, memory, and CPU requirements of the ith VM in an optimization period, respectively, and c_j^{bw}, c_j^{mem}, and c_j^{cpu} are the network bandwidth, memory, and CPU idle capacity of the jth PM, respectively.

As the cloud data center consists of a large number of PMs, the above-mentioned optimization problem is an NP-hard problem. The problem of finding the optimal target PMs is considered to be an optimization problem in which the overall transmission overhead must be minimized while satisfying all the constraints given by (4), (5), (6). Next, we introduce an adaptive heuristic algorithm based on the improved PSO algorithm to solve the optimization problem of identifying the optimal target PMs.

4.3 PSO-Based PM Selection Optimization

PSO [25] is widely used to solve a variety of optimization problems. First, it generates a group of random particles. Each particle, which represents a feasible solution and includes two parameters, i.e., velocity and position, flies in the multidimension search space at a specified velocity while referring to the best local position X_{Lbesti} and the best global positionX_{gbest}, and updates its velocity and position to move the swarm toward the best solutions as follows:

$$V_i^{t+1} = \omega V_i^t + c_1 r_1 (X_{Lbesti}(t) - X_i^t) + c_2 r_2 (X_{gbest}(t) - X_i^t), \tag{7}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1}, \tag{8}$$

where V_i^t, X_i^t, V_i^{t+1}, and X_i^{t+1}represent the velocity before the update, the position before the update, the updated velocity, and the updated position, respectively. The inertial weight coefficient ω, which linearly decreases from 0.9 to 0.4 through the search process, balances the local and global search capabilities of the particles. The positive constants c_1 and c_2, which enable the particle to learn, are referred to as cognitive learning factors, while r_1 and r_2 are random functions in the range [0, 1].

Next, PSO is adopted to solve the PM selection optimization problem. However, analysis of the specific characteristics of the PM selection optimization problem shows that the problem is a discrete optimization problem. If we want to adopt PSO to search for the optimal target PMs for the VMs hosted on the deteriorating PM, we must improve the parameters and operators of the original PSO algorithm and design the encoding scheme and fitness function.

Therefore, in the next section, we first introduce the parameters and operators of the improved PSO algorithm and then propose the encoding scheme and fitness function.

4.3.1 Parameters and Operators

Definition 1 (Subtraction operator). *The subtraction operator is represented symbolically by $\odot$, and the difference between two VM placement solutions is calculated by $x_j^t \odot x_k^t$; if the corresponding bit value of solution x_j^t is equal to that of solution x_k^t, then the corresponding bit value in the result is 1; otherwise, it is 0. For example, $(1,1,0,1) \odot (1,0,1,1) = (1,0,0,1)$.*

Definition 2 (Addition operator). *The addition operator is represented symbolically by $\oplus$, which represents the particle velocity update operation caused by its own velocity inertia, local best position, and global best position in the process of particle updating. Thus, $P_1 V_1^t \oplus P_2 V_2^t \oplus \cdots\cdots P_n V_n^t$ denotes that a particle updates its velocity using V_1^t with probability $P_1 \ldots$ and V_n^t with probability P_n. The probability value $P_i (\sum_{i=1}^{n} P_i = 1)$ is called the inertia weight coefficient; it can be calculated by (11) using the roulette wheel method. For example, 0.6 (1, 0, 0, 1) $\oplus$ 0.4(1, 0, 1, 0) = (1, 0, #, #). The probability that the value of the third bit is equal to 0 is 0.6, and the probability that its value is equal to 1 is 0.4. Since the value of the third bit is uncertain, it is denoted by #. Since the uncertain bit value influences the update of the particle velocity, its value is specified by the roulette wheel method.*

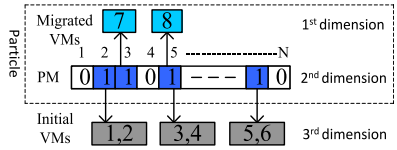


Fig. 5. Three-dimensional encoding scheme.

Definition 3 (Multiplication operator). *The multiplication operator is represented symbolically by $\otimes$, and the position update operation of the current particle position X_i^t based on the velocity vector V_k^{t+1} is denoted by $X_i^t \otimes V_k^{t+1}$. The computation rule of $\otimes$ is as follows: 1) if the corresponding bit value of the velocity vector is 1, then the corresponding bit of the position vector is not adjusted; 2) if the corresponding bit value of the velocity vector is 0, then the corresponding bit of the position vector will be re-evaluated and adjusted. For example, consider (1, 0, 1, 1) $\otimes$ (0, 1, 1, 0), where (1, 0, 1, 1) is the position vector and (0, 1, 1, 0) is the velocity. The first and fourth bit values of the velocity vector are all equal to 0, which indicates that the status of the first and fourth server in the corresponding VM placement solution should be re-evaluated and adjusted.*

Finally, the three above-mentioned definitions are used to improve the velocity updating and position updating equation of the traditional PSO [i.e., (7) and (8)] as follows, respectively [46],

$$V_i^{t+1} = P_1 V_i^t \oplus P_2 (X_{Lbesti}(t) \odot X_i^t) \oplus P_3 (X_{gbest}(t) \odot X_i^t), \tag{9}$$

$$X_i^{t+1} = X_i^t \otimes V_i^{t+1}, \tag{10}$$

where n is the length of the particle code and is equal to the number of PMs in a cloud data center, and X_i^t is an n-bit vector $(x_{i1}^t, x_{i2}^t, \ldots, x_{in}^t)$ that denotes the particle position of a feasible VM allocation solution. The value of every bit in the vector X_i^t is 0 or 1; the value is 0 if the corresponding PM is turned off and 1 otherwise. Further, V_i^t is an n-bit vector $(v_{i1}^t, v_{i2}^t, \ldots, v_{in}^t)$ that denotes the particle velocity, which represents the adjustment decisions of the VM placement. To enable VM placement such that it is an optimal solution, the above-mentioned equations are used to guide the particle position update operation. The value of every bit in the vector V_i^t is 0 or 1; the value is 0 if the corresponding PM and its VMs must be adjusted, and 1 otherwise.

4.3.2 Encoding Scheme

To solve the VM reallocation problem on the deteriorating PM, as shown in Fig. 5, we design a three-dimensional encoding scheme based on the one-to-many mapping relationship between the PMs and the VMs.

As shown in Fig. 5, the second dimension of a particle is an n-bit binary vector. Every bit in the vector is associated with a PM in a cloud data center. If the PM is active in the

current VM placement solution, the corresponding bit is 1; otherwise, it is 0. The first and third dimensions of a particle constitute a set of subsets that consist of the migrated VMs and initial VMs, respectively. Note that migrated VMs come from the deteriorating PM. Each VM subset is associated with an active PM. For example, if the fifth bit value of the second dimension of this particle is equal to 1, the fifth PM in the cloud data center should be turned on. The third, fourth, and eighth VMs should be placed onto the fifth PM, and the eighth VM is migrated from the deteriorating PM.

4.3.3 Fitness Function

To jointly execute a set of parallel applications, the VMs on the deteriorating PM and other VMs in the same virtual cluster can consume a considerable amount of network resources and a long execution time. Hence, we must minimize the overall transmission overhead to reduce the network resource consumption and the execution time. For illustration purposes, every bit in the second dimension of the particle is called the local position. The overall transmission overhead in an optimization period is called fitness, which is denoted by f_{fitness} and calculated as follows:

$$f_{fitness} = \sum_{i=1}^{m} \sum_{k=1}^{V} y_{ik} * (bw_{ki} + bw_{ik}). \tag{11}$$

When the VMs on the deteriorating PM will be allocated to other PMs in the cloud data center, our approach can select an optimal allocation solution such that f_{fitness} is minimum.

5 Performance Evaluation

In this section, we evaluate the efficiency and effectiveness of our approach through simulation experiments. Specifically, we compare our approach with five other approaches in terms of the overall transmission overhead, overall network resource consumption, and total execution time while executing a set of parallel applications.

5.1 Simulation Setup

We extend FTCloudSim simulator [13], [47], which is based on CloudSim [48], to simulate our experimental environment. All the experiments were conducted on a 16-port fat-tree data center network with 64 core switches and 16 pods. Each pod consisted of eight aggregation switches and eight edge switches. Thus, there were 128 aggregation switches and 128 edge switches in the cloud data center; each edge switch could connect to eight PMs, and each PM could host one or more VMs. In order to reflect the effect of VM reallocation, we simulated a data center comprising 1024 heterogeneous PMs and 4,000 heterogeneous VMs. Each PM was modeled to have a dual-core CPU with performance equivalent to 3,720 or 5,320 MIPS, 10 GB of RAM, 10 GB/s network bandwidth, and 1 TB of storage [49]. Each VM required one CPU core with a maximum of 360, 490, 540, 620, 720, 860, or 1,000 MIPS, 1 GB of RAM, 900 Mb/s network bandwidth, and 1 GB of storage. The capacities of the core, aggregation, and edge links were set as 10, 10, and 1 Gps, respectively. The transfer delays of the aggregation, core, and edge switches were 1, 1, and 2 s, respectively [50].

To assess the performance of the proposed approach (PCFT), we compared it with five other algorithms, namely, random first-fit (RFF), first-fit (FF), best-fit (BF), modified best fit decreasing (MBFD) [51], and IVCA. In Section 3.2, IVCA was proposed to initially allocate all VMs to the PMs of the cloud data center. However, in this section, IVCA is adopted to reallocate the VMs on the deteriorating PM to other healthy PMs in the cloud data center for comparison with four other approaches and PCFT.

In general, it is known that RFF, FF, and BF are three classical greedy approximation algorithms. When a deteriorating PM is detected, there may be multiple PM candidates that satisfy the constraints. RFF randomly selects some PMs to host the VMs on the deteriorating PM. FF always migrates the VMs on the deteriorating PM to the PMs that first meet the constraints. BF selects the PMs that achieve minimum CPU utilization for the VMs on the deteriorating PM. MBFD always moves the VMs on the deteriorating PM to the optimal PMs that can achieve the minimum transmission overhead and energy consumption.

All the above-mentioned approaches are evaluated by the following performance metrics:

- Overall transmission overhead: The overall transmission overhead between the VMs on the deteriorating PM that are migrated to the target PMs and other VMs in the same virtual cluster is calculated by (2).
- Total execution time: The total execution time for all migrated VMs and the corresponding virtual clusters to jointly execute a set of parallel applications can be calculated as follows:

$$T_{tatal} = \sum_{i=1}^{n} (T_{end}(t_i) - T_{start}(t_i)), \tag{12}$$

  where n is the number of parallel applications, and $T_{start}(t_i)$ and $T_{end}(t_i)$ are the start and end times of the ith parallel application, respectively.
- Network resource consumption: This performance metric can be evaluated by four sub-metrics, namely, $Packet_{all}$, $Packet_{root}$, $Packet_{agg}$, and $Packet_{edge}$, which can be calculated as follows:

$$Packet_{edge} = \sum_{i=1}^{n} E_i \times size(packet_i), \tag{13}$$

$$Packet_{agg} = \sum_{i=1}^{n} A_i \times size(packet_i), \tag{14}$$

$$Packet_{root} = \sum_{i=1}^{n} R_i \times size(packet_i), \tag{15}$$

$$Packet_{all} = Packet_{root} + Packet_{agg} + Packet_{edge}, \tag{16}$$

  where $Packet_{root}, Packet_{agg}, Packet_{edge}$, and $Packet_{all}$ are the total sizes of packets transferred by the root switches, aggregation switches, edge switches, and all switches, respectively. Further, R_i, A_i, and E_i are the transfer frequencies of the root switches, aggregation switches, and edge switches, respectively.

5.2 Experimental Results and Evaluation

In this section, we analyze the performance of our approach by comparing it with five other related approaches in terms

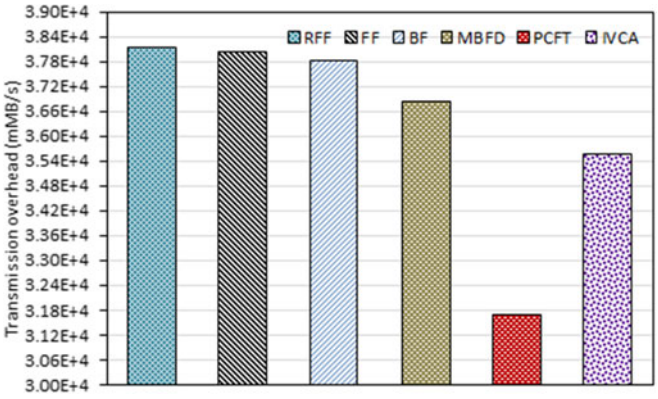


Fig. 6. Overall transmission overhead between migrated VMs and other VMs in the same virtual cluster.

of the overall transmission overhead, overall network resource consumption, and total execution time while executing a set of parallel applications.

5.2.1 Comparison of Overall Transmission Overhead

The first set of experiments aims to estimate the overall transmission overhead incurred due to migration of the VMs on the deteriorating PM to other healthy PMs. According to (2), the transmission overhead determines the execution time and network resource consumption when a virtual cluster executes a set of parallel applications.

As shown in Fig. 6, the experimental results indicate that our approach (PCFT) has the least transmission overhead compared to the other five related approaches. This is because our approach adopts the improved PSO-based approximation algorithm to search for the optimal PMs for the VMs, when the current PM is deteriorating. Thus, when these VMs are reallocated to healthy PMs, the transmission overhead is minimum. Other related approaches do not adopt a heuristic algorithm. RFF, FF, and BF have nearly similar (higher) transmission overhead, because these approaches do not consider the transmission overhead during the search of the healthy PMs, when the VMs are on the deteriorating PM. In contrast, both MBFD and IVCA consider the transmission overhead. Hence, their transmission overhead is lower as compared to RFF, FF, and BF but higher as compared to PCFT. IVCA has lower transmission overhead than MBFD because IVCA first considers the transmission overhead of the VMs when they are on the deteriorating PM. However, MBFD considers both the transmission overheads.

5.2.2 Analysis of Cloud Service Reliability Enhancement

We modeled CPU temperature to predict a deteriorating PM in order to preemptively reallocate VMs from the deteriorating PM to a healthy PM; the proactive mechanism can enhance cloud service reliability to a certain extent. We also know that the transmission overhead determines the execution time and network resource consumption when virtual clusters jointly execute a set of parallel applications. Next, we analyzed the performance of cloud service reliability enhancement on the basis of the total execution time and network resource consumption. The results are shown in Figs. 7 and 8.

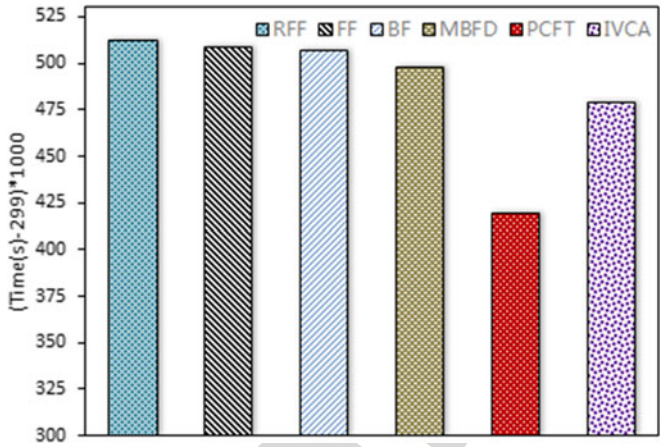


Fig. 7. Total execution time of the RFF, FF, BF, MBFD, PCFT, and IVCA approaches while executing parallel applications.

Fig. 7 shows the total execution time of the RFF, FF, BF, MBFD, PCFT, and IVCA approaches while executing a set of parallel applications. The results indicate that the total execution time of PCFT is shorter than that of the other five approaches. This is mainly because PCFT places the VMs in the same virtual cluster in a more concentrated manner than the other approaches. More precisely, PCFT needs more aggregation and edge layer switches and fewer root layer switches than the other five approaches. Hence, the communication traffic of the virtual clusters that use PCFT to reallocate the VMs on the deteriorating PM requires more aggregation and edge layer switches. Therefore, PCFT takes less time to transfer data packets from one VM to another VM in the same virtual cluster, which reduces the total execution time.

Next, we evaluated the network resource consumption of all the approaches. Fig. 8 shows the network resource consumption of edge layer switches, aggregation layer switches, core layer switches, and all layer switches, respectively. PCFT consumes the least edge layer, aggregation layer, core layer, and overall network resources as compared to the other related approaches. This is because PCFT adopts the PSO-based allocation approach to reallocate the VMs on the deteriorating PM to healthy PMs, which leads to the least transmission overhead between one VM and other VMs in the same virtual cluster. Hence, the VMs on the deteriorating PM and other VMs in the same virtual cluster are placed most likely in the same subnet or pod. In contrast, the other

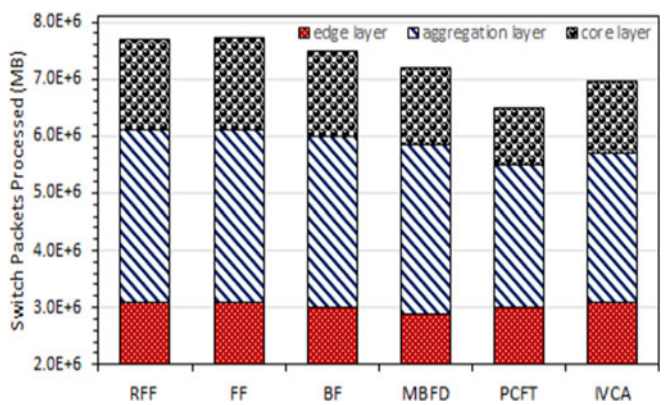


Fig. 8. Network resource consumption of the RFF, FF, BF, MBFD, PCFT, and IVCA approaches for all layers.

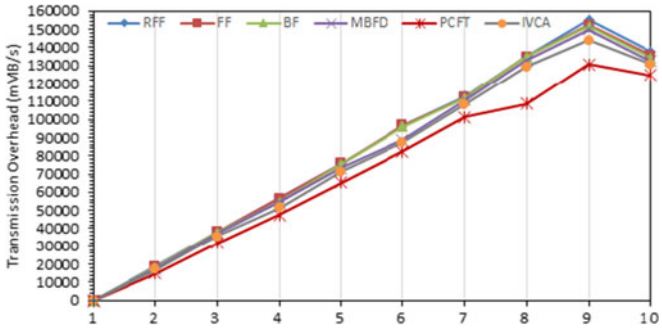


Fig. 9. Effect of virtual cluster size. The virtual cluster size represents the number of VMs in a virtual cluster. The transmission overhead of all approaches tends to increase as a whole when the value of the virtual cluster size increases from 1 to 10, and our proposed approach (PCFT) has the slowest growth rate.

five approaches place the VMs in a more dispersed manner as compared to PCFT. As a result, more core layer switches are utilized. As per the fat-tree architecture, all packets routed through the core layer switches will also be transferred by the aggregation and edge layer switches. Thus, the core link becomes congested easily. Consequently, we should try to reduce the network resource consumption of the core link to enhance cloud service reliability.

From the experimental results, we can conclude that PCFT outperforms the other five related approaches. Moreover, it demonstrates the same effect on cloud service reliability enhancement as the related approaches.

5.3 Study of Parameters

In this section, we study the effect of the experimental parameters on all the approaches. As shown in Figs. 9, 10, and 11, the parameters include the virtual cluster size, number of parallel applications, and number of VMs. In our experiments, the virtual cluster size was set at 3, the number of VMs was set at 4,000, and the number of parallel applications was set at 1,000.

5.3.1 Effect of Virtual Cluster Size

Fig. 9 shows the effect of the virtual cluster size on all the approaches. To clearly show its impact, the number of VMs was set at 4,000, and the number of parallel applications was set at 1,000. We varied the value of the virtual cluster size from 1 to 10 in steps of 1 in the experiment. The figure shows that the transmission overhead of all the approaches tends to increase as a whole, and the growth rate of PCFT is the lowest. We designed a parallel application model executed by a virtual cluster including three VMs. Experimental results of overall network resource consumption and total execution time under other virtual cluster sizes have not been provided. However, owing to the relationship between the transmission overhead and other performance metrics, we believe that with an increase in the virtual cluster size, the total execution time and overall network resource consumption will be affected to some extent. Further, the cloud service reliability is also affected to some extent.

5.3.2 Effect of Number of Parallel Applications

Fig. 10 shows the effect of the number of parallel applications on all the approaches. To clearly show its impact, the

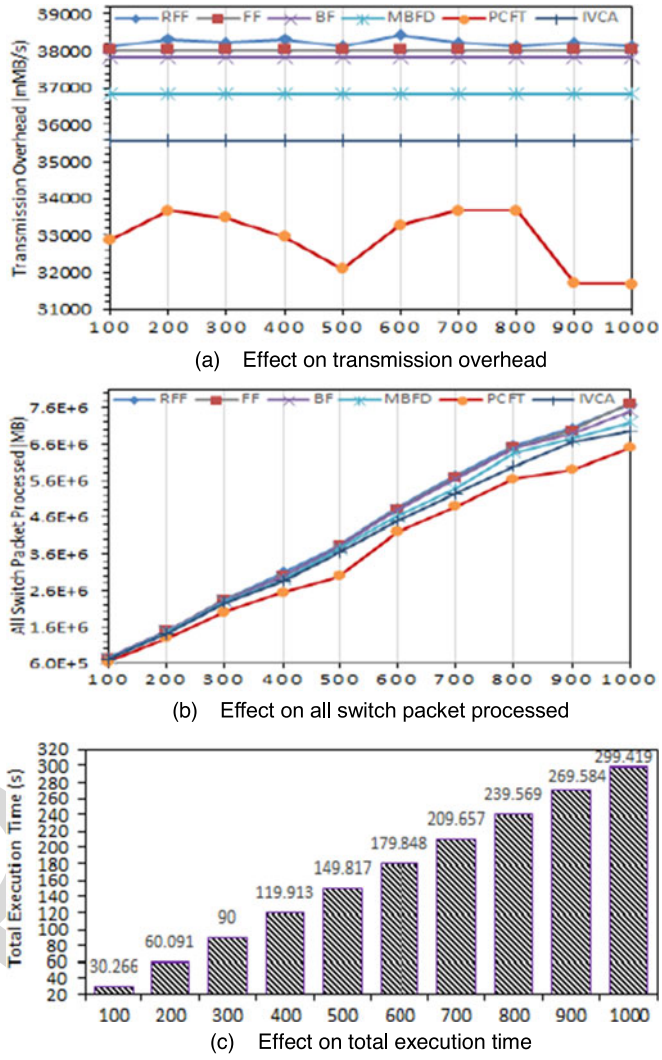
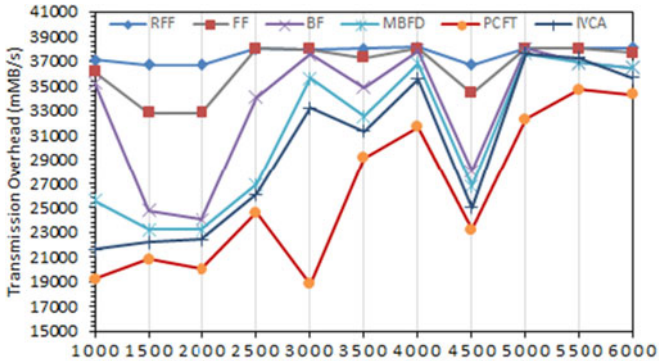


Fig. 10. Effect of number of parallel applications. The number of parallel applications represents the number of tasks processed. The transmission overhead of our approach (PCFT) is not affected significantly by this parameter. All packets routed through all the three layer switches and the total execution time of the parallel applications increase with the number of parallel applications. The PCFT approach has the lowest network resource consumption growth rate. (a) Effect on transmission overhead, (b) effect on all switch packet processed, and (c) effect on total execution time.

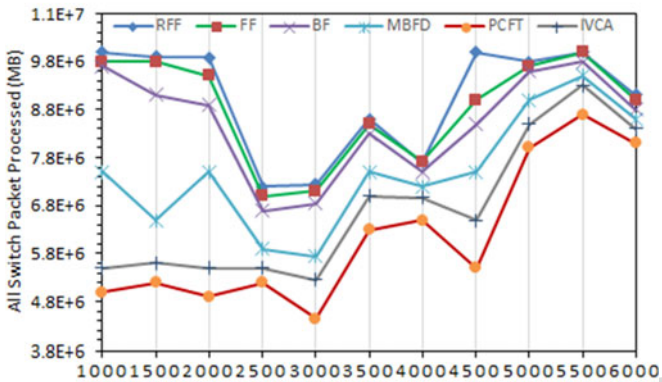
virtual cluster size was set at 3, and the number of VMs was set at 4,000. We varied the number of parallel applications from 100 to 1,000 in steps of 100 in this experiment. The figure indicates that the transmission overhead is not affected significantly by the number of parallel applications; however, the overall network resource consumption and the total execution time steadily increase with the number of applications, and the network resource consumption growth rate of PCFT is the lowest. Further, through this observation, we can safely conclude that the cloud service reliability decreases.

5.3.3 Effect of Number of VMs

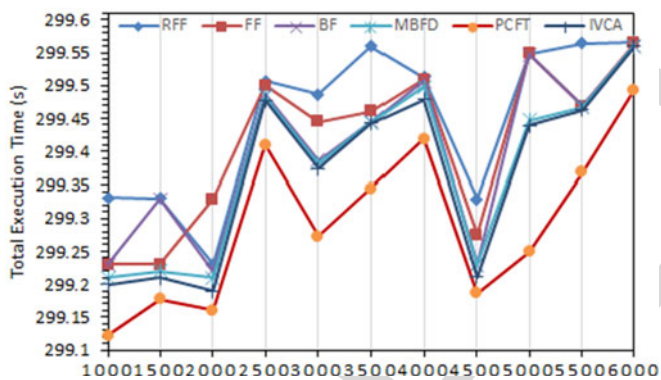
Fig. 11 shows the effect of the number of VMs on all the approaches. To clearly show its impact, the virtual cluster size was set at 3, and the number of parallel applications was set at 1,000. We varied the number of VMs from 1,000 to 6,000 in steps of 500 in this experiment. These figures



(a) Effect on transmission overhead



(b) Effect on all switch packet processed



(c) Effect on total execution time

Fig. 11. Effect of number of VMs. The number of VMs represents the number of VMs initially placed. The transmission overhead, all packets routed through all the three layer switches, and total execution time for 1000 parallel applications increase with the number of VMs, but our proposed approach (PCFT) has the lowest growth rate for each figure. (a) Effect on transmission overhead, (b) effect on all switch packet processed, and (c) Effect on total execution time.

show that the transmission overhead, all packets routed through all the three layer switches, and total execution time increase with the number of VMs, and our approach (PCFT) has the lowest growth rate. This is mainly because the total number of PMs was 1,024 in the cloud data center; each PM had a fixed hardware configuration. Thus, the parameters increased with the number of VMs. Further, based on the observations in these experiments, we can safely conclude that the cloud service reliability decreased.

6 Conclusions and Future Work

In this work, we proposed a PCFT approach that adopts a VM coordinated mechanism to anticipate a deteriorating PM in a cloud data center, and then automatically migrates VMs from the deteriorating PM to the optimal target PMs. This is a very challenging problem, considering its efficiency, effectiveness, and scalability requirements. We addressed this problem through a two-step approach, where we first proposed a CPU temperature model to anticipate a deteriorating PM, and then searched for the optimal target PMs using an efficient heuristic algorithm. We evaluated the performance of the PCFT approach by comparing it with five related approaches in terms of the overall transmission overhead, overall network resource consumption, and total execution time while executing a set of parallel applications. The experimental results demonstrated that our proposed approach outperforms the other five related approaches.

In our experiments, for ease of understanding, we designed a set of parallel applications, where each parallel application consists of three tasks, in order to validate our approach. However, complex parallel applications still need to be designed in our experimental platform. Hence, in the future, we will design multiple types of parallel applications for execution in our experimental platform. Meanwhile, we also plan to apply our approach to reactive FT using the full coordinated checkpoint mechanism, which helps to reduce checkpoint frequencies as fewer unanticipated failures are encountered, in addition to reducing network and storage resource consumption while guaranteeing cloud service reliability.

Acknowledgments

This work is supported by the NSFC (61272521 and 61472047).

References

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proc. 9th IEEE Grid Computing Environments Workshop*, 2008, pp. 1–10.

[2] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[3] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proc. 10th ACM Comput. Commun. Rev.*, 2011, pp. 350–361.

[4] K. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *Proc. 1st ACM Symp. Cloud Comput.*, 2010, pp. 193–204.

[5] M. Schwarzkopf, D. Murray, and S. Hand, "The seven deadly sins of cloud computing research," in *Proc. 4th USENIX Workshop Hot Topics Cloud Comput.*, Jun. 2012, p. 1.

[6] M. Treaster, "A survey of fault-tolerance and fault-recovery techniques in parallel systems," in *Proc. 5th ACM Comput. Res. Repository*, Jan. 2005, pp. 1–11.

[7] R. Jhawar and V. Piuri, "Fault tolerance and resilience in cloud computing environments," in *Computer and Information Security Handbook*, 2014, pp. 1–28. Q1

[8] G. Jung, K. Joshi, M. Hiltunen, R. Schlichting, and C. Pu, "Performance and availability aware regeneration for cloud based multitier applications," in *Proc. 40th IEEE/IFIP Dependable Syst. Netw.*, 2010, pp. 497–506.

[9] Z. Zheng, T. Zhou, M. Lyu, and I. King, "Component ranking for faulttolerant cloud applications," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 540–550, 4th Quarter, 2012.

[10] Z. Zheng, T. Zhou, M. Lyu, and I. King, "FTCloud: A rank ingBased framework for fault tolerant cloud applications," in *Proc. 21th IEEE Int. Symp. Softw. Rel. Eng.*, 2010, pp. 398–407.

[11] R. Koo and S. Toueg, "Checkpointing and rollback-recovery for distributed systems," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 1, pp. 25–31, Jan. 1987.
[12] I. Goiri, F. Julia, J. Guitart, and J. Torres, "Checkpoint-based fault tolerant infrastructure for virtualized service providers," in *Proc. IEEE/IFIP Netw. Operations Manag. Symp.*, 2010, pp. 455–462.
[13] A. Zhou et al., "On cloud service reliability enhancement with optimal resource usage," *IEEE Trans. Cloud Comput.*, vol. PP, pp. 1–1, 2014.
[14] C. Coti et al., "Blocking vs. non-blocking coordinated checkpointing for large-scale fault tolerant MPI," in *Proc. 19th ACM/IEEE Conf. Supercomput.*, 2006, pp. 11–20.
[15] K. Chandy and L. Lamport, "Distributed snapshots: Determining global states of distributed systems," *ACM Trans. Comput. Syst.*, vol. 3, no. 1, pp. 63–75, 1985.
[16] M. Zhang, H. Jin, X. Shi, and S. Wu., "Virtcft: A transparent vmlevel fault-tolerant system for virtual clusters," in *Proc. 16th IEEE Int. Conf. Parallel Distrib. Syst.*, 2010, pp. 147–154.
[17] B. Cully et al., "Remus: High availability via asynchronous virtual machine replication," in *Proc. 5th USENIX Symp. Netw. Syst. Des. Implementation* , 2008, pp. 161–174.
[18] A. Nagarajan, F. Mueller, C. Engelmann, and S. Scott, "Proactive fault tolerance for HPC with Xen virtualization," in *Proc. 21th Int. Conf. Supercomput.*, 2007, pp. 23–32.
[19] P. Barham et al., "Xen and the art of virtualization," in *Proc. 19th ACM Symp. Operating Syst. Principles*, 2003, pp. 164–177.
[20] M. Dong, H. Li, K. Ota, L. T. Yang, and H. Zhu, "Multicloud-based evacuation services for emergency management," *IEEE Cloud Comput.*, vol. 1, no. 4, pp. 50–59, Nov. 2014.
[21] J. Ho, P. Hsiu, and M. Chen, "Improving serviceability for virtual clusters in bandwidth-constrained datacenters," in *Proc. 8th IEEE Int. Conf. Cloud Comput.*, 2015, pp. 710–717.
[22] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *Proc. ACM Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2011, pp. 98–109.
[23] Cisco Global Cloud Index, "Forecast and Methodology, 2013–2018," [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf, 2014.
[24] D. Breitgand and A. Epstein, "Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds," in *Proc. 31th IEEE Int. Conf. Comput. Commun.*, 2012, pp. 2861–2865.
[25] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.
[26] F. Machida, M. Kawato, and Y. Maeno, "Redundant virtual ma chine placement for fault-tolerant consolidated server clusters," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp.*, 2010, pp. 32–39.
[27] E. Bin, O. Biran, O. Boni, E. Hadad, E. Kolodner, Y. Moatti, and D. Lorenz, "Guaranteeing high availability goals for virtual machine placement," in *Proc. 31th Int. Conf. Distributed Comput. Syst.*, May 2011, pp. 700–709.
[28] S. Deng, L. Huang, J. Taheri, and A. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Trans. Parallel Distributed Syst.*, vol. 26, no. 12, pp. 3317–3329, 2015.
[29] S. Wang, A. Zhou, C. Hsu, X. Xiao, and F. Yang, "Provision of data-intensive services through energy-and QoS-aware virtual machine placement in National Cloud Data Centers," *IEEE Trans. Emerging Topics Comput.*, vol. PP, no. 99, pp. 1–14, 2015.
[30] Z. Zheng, Y. Zhang, and M. Lyu, "CloudRank: A QoS-Driven component ranking framework for cloud computing," in *Proc. 29th IEEE Int. Symp. Reliable Distributed Syst.*, 2010, pp. 184–193.
[31] H. Li, M. Dong, X. Liao, and H. Jin, "Deduplication-based energy efficient storage system in cloud environment," *Comput. J.*, vol. 58, no. 6, pp. 1373–1383, 2015.
[32] E. N. Elnozahy, L. Alvisi, Y. M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message-passing systems," *ACM Comput. Surveys*, vol. 34, no. 3, pp. 375–408, 2002.
[33] A. Ciuffoletti, "Error recovery in systems of communicating processes," in *Proc. 7th Int. Conf. Softw. Eng.*, 1984, pp. 6–17.
[34] S. Yi, D. Kondo, and A. Andrzejak, "Reducing costs of spot in stances via checkpointing in the amazon elastic compute cloud," in *Proc. 3th IEEE Int. Conf. Cloud Comput.*, Jun. 2010, pp. 236–243.
[35] Y. Liu, et al., "An optimal checkpoint/restart model for a large scale high performance computing system," in *Proc. 22th IEEE Int. Symp. Parallel Distrib. Process.*, 2011, pp. 1–9.
[36] N. Limrungsi et al., "Providing reliability as an elastic service in cloud computing," in *Proc. IEEE Int. Conf. Commun.*, 2012, pp. 2912–2917.
[37] X. Liu, Y. Ma, Y. Liu, T. Xie, and G. Huang, "Demystifying the imperfect client-side cache performance of mobile web browsing," *IEEE Trans. Mobile Comput.*, vol. PP, no. 99, pp. 1536–1233, 2016.
[38] S. Garg and R. Buyya, "An environment for modeling and simulation of message-passing parallel applications for cloud computing," *Softw.: Practice Experience*, vol. 43, no. 11, pp. 1359–1375, 2013.
[39] S. Garg and R. Buyya, "Networkcloudsim: Modelling parallel applications in cloud simulations," in *Proc. 4th IEEE Int. Conf. Utility Cloud Comput.*, 2011, pp. 105–113.
[40] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM Comput. Commun. Rev.*, 2008, pp. 63–74.
[41] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proc. 6th IEEE/ACM Int. Conf. Green Comput. Commun.*, 2010, pp. 179–188.
[42] T. Heath et al., "Mercury and Freon: Temperature emulation and management for server systems," *Proc. 12th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2006, pp. 106–116.
[43] L. Ramos and R. Bianchini, "C-Oracle: Predictive thermal man agement for data centers," in *Proc. 14th IEEE Int. Symp. High Perform. Comput. Archit.*, 2008, pp. 111–122.
[44] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. 9th ACM Int. Conf. Internet Meas.*, 2009, pp. 202–208.
[45] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proc. 30th IEEE Int. Conf. Comput. Commun.*, 2011, pp. 71–75.
[46] S. Wang, Z. Liu, Z. Zheng, Q. Sun, and F. Yang, "Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers," in *Proc. 19th IEEE Int. Conf. Parallel Distrib. Syst.*, 2013, pp. 102–109.
[47] A. Zhou, S. Wang, Q. Sun, H. Zou, and F. Yang, "FTCloudSim: A simulation tool for cloud service reliability enhancement mechanisms," in *Proc. 14th ACM/IFIP/USENIX Int. Middleware Conf. Demo Poster Track*, 2013, pp. 1–2.
[48] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw.: Practice Experience*, vol. 41, 1, pp. 23–50, 2011.
[49] A. Beloglazov and R. Buyya, "Optimal online deterministic algo rithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency Comput.: Practice Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.
[50] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue, "Survivable virtual infrastructure mapping in virtualized data centers," in *Proc. 5th IEEE Int. Conf. Cloud Comput.*, Jun. 2012, pp. 196–203.
[51] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware re source allocation heuristics for efficient management of data centers for Cloud computing," *Future Green Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

Jialei Liu received the ME degree in computer science and technology from Henan Polytechnic University in 2008. He is currently working toward the PhD degree at Beijing University of Posts and Telecommunications, Beijing, China. His research interests include cloud computing and service reliability.

Shangguang Wang received the PhD degree from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2011. He is an Associate Professor at the State Key Laboratory of Networking and Switching Technology, BUPT. He is the Vice Chair of IEEE Computer Society Technical Committee on Services Computing, President of the Service Society Young Scientist Forum in China and served as the General Chair of CollaborateCom 2016, General Chair of ICCSA 2016, TPC Chair of IOV 2014, and TPC Chair of SC2 2014. His research interests include service computing, cloud computing, and QoS Management. He is a Senior Member of the IEEE.

Ao Zhou received the ME degree in computer science and technology from Beijing University of Posts and Telecommunications, Beijing, China, in 2012. She is currently working toward the PhD degree at Beijing University of Posts and Telecommunications. Her research interests include cloud computing and service reliability.

Sathish A. P. Kumar received the PhD degree in computer science and engineering from the University of Louisville, KY, in 2007. He is currently an Assistant professor at the Coastal Carolina University, SC, USA. He has published more than 30 papers. His current research interests include cloud computing security and reliability and service computing. He is a senior member of the IEEE.

Fangchun Yang received the PhD degree in communications and electronic systems from the Beijing University of Posts and Telecommunication, Beijing, China, in 1990. He is currently a professor at the Beijing University of Posts and Telecommunication, China. He has published six books and more than 80 papers. His current research interests include network intelligence, service computing, communications software, soft-switching technology, and network security. He is a fellow of the IET.

Rajkumar Buyya a professor of computer science and software engineering, future fellow of the Australian Research Council, and the director in the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in cloud computing. He has authored over 450 publications and five textbooks including *Mastering Cloud Computing* published by McGraw Hill and Elsevier/Morgan Kaufmann, 2013 for Indian and international markets, respectively. Software technologies for grid and cloud computing developed under his leadership have gained rapid acceptance and are in use at several academic institutions and commercial enterprises in 40 countries around the world. He has led the establishment and development of key community activities, including serving as the foundation Chair in the IEEE Technical Committee on Scalable Computing and five IEEE/ACM conferences. These contributions and international research leadership of him are recognized through the award of "2009 IEEE TCSC Medal for Excellence in Scalable Computing." He is currently serving as co-editor-in-chief of *Journal of Software: Practice and Experience*, which was established 40+ years ago. He is a fellow of the IEEE.

Queries to the Author

Q1. Please provide publisher name and its location in Ref. [7].
Q2. Please update Refs. [13], [29], and [37].
Q3. CE: Please check volume