A Heuristic Time Sharing Policy for Backup Resources in Cloud System

Xinyi Li, Yong Qi, Pengfei Chenand Xiaohui Zhang
Department of Computer Science and Technology, Xi'an Jiaotong University
Xi'an, Shaanxi 710049 - China
[e-mail: xingga.li@stu.xjtu.edu.cn]
*Corresponding author: Xinyi Li

Received February 10, 2016; revised May 15, 2016; accepted May 22, 2016; published July 31, 2016

Abstract

Cloud computing promises high performance and cost-efficiency. However, most cloud infrastructures operate at a low utilization, which greatly adheres cost effectiveness. Previous works focus on seeking efficient virtual machine (VM) consolidation strategies to increase the utilization of virtual resources in production environment, but overlook the under-utilization of backup virtual resources. We propose a heuristic time sharing policy of backup VMs derived from the restless multi-armed bandit problem. The proposed policy achieves increasing backup virtual resources utilization and providing high availability. Both the results in simulation and prototype system experiments show that the traditional 1:1 backup provision can be extended to 1:M (M≫1) between the backup VMs and the service VMs, and the utilization of backup VMs can be enhanced significantly.

Keywords: Cloud computing, virtualization, availability, restless multi-armed bandit, back up

This work was supported in part by the National Natural Science Foundation of China under Grant No.61272 460 and the Specialized Research Fund for the Doctoral Program of Higher Education under Grant No. 20120 201110010.

http://dx.doi.org/10.3837/tiis.2016.07.009 **ISSN : 1976-7277**

1. Introduction

An increasing amount of computing is now shifted to the public clouds such as Amazon's EC2 [1], Windows Azure [2], and Google Compute Engine [3], or to private clouds managed by frameworks such as VMware vCloud [4] and OpenStack [5]. While appreciating the expediency in the cloud environment, complex bugs rooted in software stacks, continually changing environment and random hardware failures could lead to increasing difficulties in availability assurance. Failures and their impacts on system performance and operation costs are becoming an increasingly important concern for system designers and administrators [6-7]. According to Cerin's study [8], software failures have led up to tens of millions of dollars loss in current mainstream cloud systems including Amazon, Windows Azure, Google App, etc. In order to ensure the agreed level of availability in the cloud, providing backup resources is commonly adopted by cloud operators [9]. Even though the weakest fault-tolerant methodology called 1-resilient virtual machine (VM) [10] is employed, additional n VMs will be deployed simultaneously when there are n service VMs in the cloud system. However, excessive backup resources leads to underutilized computing resources as well as high operation and maintenance cost in cloud infrastructures. The overpaid equipment procurement and energy expenditure cannot be neglected. Both researchers and engineers in academia and industry focus on seeking efficient VM deployment and VM consolidation strategies in order to reduce the capital cost in the cloud platform. For instance, Ahn et al. [11] proposed a novel auto-scaling mechanism which can automatically allocate virtual resources on an on-demand basis in real-time healthcare applications. However, the underutilization of backup virtual resources are ignored.

Previous researches indicate that failures are sparse for a single service VM [12], and the corresponding backup VM is idle in most of the time. Although handling these rare failures is extremely important for ensuring the availability of software, the associated high cost for providing a backup VM for each service VM may not be necessary. Provisioning cost effective backup infrastructures in the cloud system based on these failure characteristics poses interesting research challenges.

This paper brings forth a novel idea called time sharing of backup VMs. We try to utilize a backup VM to provide backup service for multiple failed service VMs. Thus, we can reduce the number of deployed backup VMs, the required power provisioning and the maintenance expenses. By improving the backup resources utilization through time sharing, we reduce the total cost of ownership (TCO) for building and operating the cloud platform.

However, sharing backup VMs is very complicated. Firstly, there are a large number of, maybe thousands of, backup VMs in a large-scale cloud system. Establishing a behavior model for each backup VM will eventually result in a super high dimensional problem, which is well known to be intractable due to the exponential computation complexity. Secondly, in the cloud system, multiple service VMs may fail simultaneously at any

moment. Selecting as many idle backup VMs as possible to provide backup services is worthy of thinking carefully. Thirdly, due to dynamics of service VM failures, on-line and real-time scheduling strategy is of necessity. Fourthly, we must guarantee that the time sharing of backup VMs would not cause significant decrease in terms of system availability. Therefore when the number of backup VMs is fixed, figuring out the upper bound of service VMs which can be scheduled to the backup VMs is also a difficulty.

To solve this challenging problem, we first formalize the time sharing problem of backup VMs as a two-state restless multi-armed bandit (RMAB) process. Then, the intractable n-dimensional optimization problem is decomposed into n one-dimensional problems by leveraging a heuristic policy. Based on the VM failure and restoration models, a heuristic time sharing policy is developed to tackle both homogeneous and heterogeneous cases. In the homogeneous case, each service VM evolves as an independent and identically distributed Markov process, while in the heterogeneous case, each service VM evolves as an independent and non-identically distributed Markov process. Through the validation in simulation experiments, the proposed policy extends the traditional 1:1 backup provision to 1:M ($M \gg 1$) between the backup VM and the service VMs with guaranteed backup of failed service VMs to ensure the system availability and the utilization of backup resources can be increased significantly.

This paper is organized as follows. Section 2 describes the background information and the motivation. The time sharing backup VMs problem is formulated in Section 3. Section 4 presents the heuristic time sharing policy of backup VMs and proves the optimality conditions of the proposed policy. Section 5 shows the evaluation results of the heuristic time sharing policy in simulation experiment and a prototype system. Section 6 presents the related work. Section 7 concludes this paper.

2. Background and Motivation

Before the detailed description of the problem and policy developed in this paper, this section introduces the failure and restoration models of VMs, RMAB problem and motivation.

2.1 Failure and Restoration Model

During the lifetime of a software system, failures occur due to several reasons including software bugs, hardware failures, power outage, external environment changes, network jam and human operations [13]. Features of software failures and restorations have so long been an open topic in the field of software availability. Researchers view the sequence of failure events as a stochastic process and study the time distribution between failures [14]. Another important factor affecting software availability is the restoration time. It has been shown that the time between failures is modeled well by Weibull distribution [12,15] and Poisson distribution [16-18], and the restoration time is well modeled by Lognormal distribution [12]. In order to be applied to various failure and restoration scenarios, this paper discusses the situations in both homogeneous and heterogeneous cases. In the

homogeneous case, the states of each service VM evolve according to the same failure model and restoration model, while in the heterogeneous case, the states of service VMs evolve according to different distribution models.

2.2 RMAB

The classic multi-armed bandit (MAB) problem came from gamble games, originally proposed by Robbins [19] in 1952. It can be seen as a sequential resource allocation problem. In the standard MAB problem, a player operates only one arm out of K independent arms in each time slot and obtains a reward determined by the state of the operated arm. Only the operated arm changes its state, while the states of the unoperated arms remain frozen. It was Whittle who first suggested extending the concept of the traditional MAB problem and proposed the concept of RMAB by allowing the state of a bandit to evolve even if the bandit is not operated [20]. In recent years, RMAB problem is further extended and applied in a broad range of applications. For example, RMAB is employed to design an optimal channel sensing policy in cognitive radio networks [21-24], to make an optimal investment strategy in financial investment [25] and to schedule an optimal target tracking in modern radar systems [26].We employ the stochastic optimization model based on RMAB to formulate the time sharing of backup VMs. This paper studies the optimal allocation of backup virtual resources in the cloud system based on RMAB.

2.3 Motivation

The strategy for the time sharing of backup VMs is fundamentally based on two facts, namely the sparse failures of a single VM and the dense failures of a large number of VMs. The cumulative distribution of time between failures is shown in **Fig. 1** [12]. It can be observed from **Fig. 1** that 80% of the times between failures at a node fall in the range of 3-300 hours. Because there are thousands of nodes in a cloud system, failures become frequent in each time slot. Thus, if a backup VM is shared by multiple failed service VMs

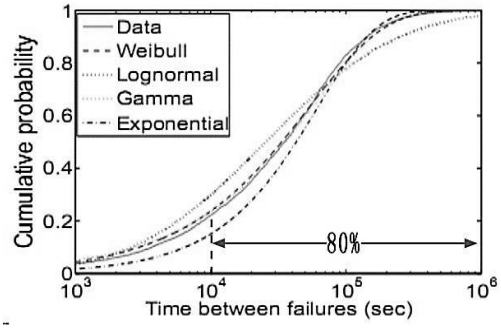


Fig. 1. The cumulative distribution of time between failures cited from [12]

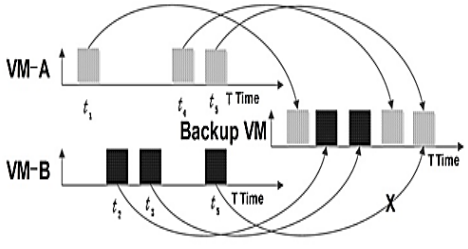


Fig. 2. The basic idea of time sharing of backup VMs

during its idle periods, the utilization of backup resources can be significantly improved. The basic idea of the time sharing of backup VMs is presented in **Fig. 2**. Backup VM is shared by VM-A and VM-B. During the time slots $t_1 - t_4$, VM-A and VM-B can efficiently share Backup VM, while in time slot t_5, VM-B will be invalid because VM-A is dispatched to Backup VM. Thus, the essence of time sharing of backup VMs is to mandate studying the backup assurance of failed service VMs in order to guarantee the system availability. In this paper, we propose a heuristic policy to select as many free backup VMs as possible for failed service VMs in each time slot, and to meet availability constraints with the limited number of backup VMs.

3. Problem Formulation

In the cloud system, some service VMs running applications with the requirement of super high availability are provided with dedicated backup VMs, while other service VMs without the requirement of super high availability are not provided with dedicated backup VMs. When a backup VM is not occupied by its dedicated service VM, it can be used by a failed service VM, which is called time sharing of backup VMs.

A backup VM is capable of providing backup service for several service VMs in different time periods when multiple applications are deployed on it. The time sharing of backup VMs can be achieved by starting a specific application on demand. For instance, a backup VM hosts both Tomcat and Apache HTTPD services. When a service VM running Tomcat fails, new requests can be delivered to the backup VM using request redirection technique. Ongoing Apache HTTPD service can be achieved in a similar way when the backup is needed. Deploying all the services on a single backup VM is impossible because there are a large number of various services in a cloud system. Hence, we propose a concept of "shadow backup VM". What sharing backup VMs means is to share the shadow backup VMs.

Definition 1 (Shadow backup VM): A shadow backup VM is defined as the reserved physical resources for the backup purpose of specific VMs.

A shadow backup VM is a bulk of physical resources which are able to accommodate with specific VMs. For example, a shadow backup VM is configured with 2 vcpu, 1GB memory and 300Mbps bandwidth. Then this backup VM can work for service VMs with the configuration of CPU≤2 vcpu, memory≤1GB and bandwidth≤300Mbps. In this paper, we solely address the sharing of identical shadow backup VMs.

Two more definitions are proposed for explicit depiction of the problem formulation.

Definition 2 (Primary VM): The primary VM represents the service VM with a dedicated backup VM.

Definition 3 (Secondary VM): The secondary VM represents the service VM without any dedicated backup VM and can occupy an idle backup VM which is dedicated to a primary VM.

Markov-based models are frequently adopted to analyze the availability of complex systems [27]. This kind of model usually recognizes the system as a stochastic process with several states. These states are generally classified into two groups according to the

degradation degree of a system, namely a group of normal states in which the system is available and a group of failure states in which the system is unavailable [28]. With a Markov-based model, some close-form analytical results could be obtained. Based on this model, we formulate the software system as a simple two-state Discrete Time Markov Chain.

At the beginning, N primary VMs are allocated with N backup VMs in the form of 1:1 mapping. Thus the availability of the N primary VMs is guaranteed. There are two states, "normal" (1) and "failure" (0), for each primary VM. There are also two states, "idle" (1) and "busy" (0), for each backup VM. The states of the N backup VMs can directly reflect the running status of the N primary VMs. When a backup VM is idle, the corresponding primary VM is in the normal state; otherwise, when a backup VM is busy, the corresponding primary VM is in the failure state. The idea of time sharing of backup VMs allows a failed secondary VM to use a backup VM which is not occupied by its dedicated primary VM. In each time slot, a backup VM is selected only when a backup VM is idle, i.e., the corresponding primary VM stays in the normal state. Thus, a two-state Markov model is sufficient for modeling the software system from the viewpoint of backup VM utilization. This simple two-state Markov chain is reasonable and strongly supported by employing the "belief state" [29] which is detailed in Section 4.1. Moreover, the proposed two-state Markov process makes it possible to analyze the VM backup problems mathematically.

Let $s_j(t)$ denote the state of backup VM j in time slot t, where $j \in \{1,2,\dots,N\}$. Then $S(t) = \{s_j(t), j \in \{1,2,\dots,N\}\}$ is the set of states of the N backup VMs in time slot t. Each backup VM evolves as an independently distributed two-state discrete-time Markov chain as showed in **Fig. 3**. The state-transition matrix of backup VM j is represented by P_j as:

$$P_j = \begin{bmatrix} p_{00}^j & p_{01}^j \\ p_{10}^j & p_{11}^j \end{bmatrix} = \begin{bmatrix} p_{00}^j & 1-p_{00}^j \\ p_{10}^j & 1-p_{10}^j \end{bmatrix} \quad (1)$$

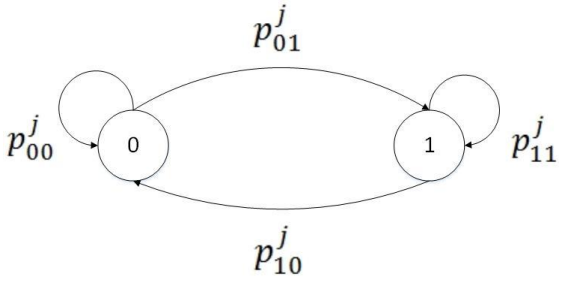


Fig. 3. The state-transition of backup VMs

In each time slot, some backup VMs are selected. Then live migration is employed to schedule failed secondary VMs to the selected idle backup VMs. Considering limited computing resources such as CPU and network bandwidth, the number of selected backup

VMs is constrained in each time slot. It is assumed that only $K(1 \le K \le N)$ backup VMs can be selected in each time slot. Let $A(t)$ denote the decision policy in time slot t, which is a set of selected backup VMs and $|A(t)| = K$. Under the decision policy $A(t)$, a reward $R(S(t))$ can be obtained. We define $R(S(t))$ as $R\big(S(t)\big) = \sum_{j \in A(t)} s_j(t)$. When a selected backup VM is in state "1", one unit of reward is gained and a failed secondary VM can occupy the idle backup VM. Otherwise, no reward can be obtained.

When the system is running in a steady state, primary VM j falls into the "normal-failure" circulation process and its dedicated backup VM falls into the "idle-busy" circulation process correspondingly. In a cycle, the times spent on states "1" and "0" of backup VM j are defined as:

$$T_1^j = E\big(D_f^j\big), T_0^j = E\big(D_r^j\big) \tag{2}$$

where D_f^j and D_r^j denote the probability distributions of the time lengths that primary VM j is in the normal and failure states, respectively, in a cycle. E(·) represents the expectation operator.

The times spent on state "1" or "0" can be discretized into a series of "1" or "0", as shown in **Fig. 4**. Each "1" or "0" means backup VM j is in state "1" or "0" in the corresponding time slot.

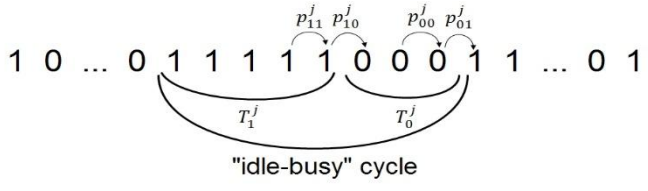


Fig. 4. Discretized representation of backup VM "idle-busy" cycle

Furthermore, the probabilities in states "1" and "0" can be given by:

$$\pi_1^j = \frac{T_1^j}{T_0^j + T_1^j},\ \pi_0^j = \frac{T_0^j}{T_0^j + T_1^j} \tag{3}$$

The state-transition matrix of backup VM j in the steady state can be given by:

$$P_j = \begin{bmatrix} p_{00}^j & p_{01}^j \\ p_{10}^j & p_{11}^j \end{bmatrix} = \begin{bmatrix} \frac{T_0^j - 1}{T_0^j} & \frac{1}{T_0^j} \\ \frac{1}{T_1^j} & \frac{T_1^j - 1}{T_1^j} \end{bmatrix} \tag{4}$$

According to the Markov steady state equation, namely $SP = S$, the probabilities of backup VM j in states "1" and "0" are

$$S_1^j = \frac{T_1^j}{T_0^j + T_1^j} = \pi_1^j,\ S_0^j = \frac{T_0^j}{T_0^j + T_1^j} = \pi_0^j \tag{5}$$

The decision policy π over T time slots is given as $\pi = \{A(t), t \in \{1,2,\dots,T\}\}$. The main purpose in this paper is to find the optimal decision policy π^* that maximizes the total expected discounted reward over T time slots. The following equation gives the formal definition of the optimal decision policy:

$$\pi^* = arg\max_{\pi} \mathrm{E}[\sum_{t=1}^{T} R_\pi(S(t))] \tag{6}$$

where $R_\pi(S(t))$ is the reward collected in time slot t under the decision policy π.

4. Problem Solving

In this section, we first present our simple and effective heuristic time sharing policy. Then the optimality in both homogeneous and heterogeneous cases is proved. At last, we briefly analyze the complexity of the proposed policy.

4.1 Heuristic Time Sharing Policy

In each time slot, as only K out of N backup VMs can be selected and observed, the state information of all the backup VMs cannot be obtained completely. Hence, the decision policy has to infer the backup VM states from its past decision and observation history so as to make its future decision. We employ the concept "belief state" [29] commonly used in partially observable Markov decision process to represent the sufficient statistical characteristic of a backup VM. The backup VM belief state vector is defined as $\Omega(t) = [\omega_1(t), \dots, \omega_N(t)]$, where $0 \le \omega_j(t) \le 1$ is the conditional probability that backup VM j is in state "1" $(s_j(t) = 1)$ in time slot t, given all past states, actions and observations. The belief state can reveal the degradation process of a software system. Specifically, the belief states of backup VMs implicitly split the VM states (i.e., {0,1}) into many continuous states (i.e., [0,1]) indicating the failure probability of the dedicated primary VMs. Due to the Markov nature of the backup VM state-transition process, the belief state of backup VM j can be updated recursively with Bayes rule as follows:

$$\omega_j(t+1) = \begin{cases} p_{01}^j & if\ j \in A(t), s_j(t) = 0 \\ p_{11}^j & if\ j \in A(t), s_j(t) = 1 \\ \tau_j\left(\omega_j(t)\right) & if\ j \notin A(t) \end{cases} \tag{7}$$

where $\tau_j\left(\omega_j(t)\right) = \omega_j(t)p_{11}^j + (1-\omega_j(t))p_{01}^j$ denotes the one-step belief state update of unselected backup VM j. If each backup VM is regarded as a bandit, the time sharing of backup VMs turns out to be a RMAB process.

The optimal decision policy π^* that maximizes the total expected discounted reward over T time slots can be formally defined as:

$$\pi^* = arg\max_{\pi} E[\sum_{t=1}^{T} R_\pi(\Omega(t))|\Omega(1)] \tag{8}$$

where $R_\pi(\Omega(t))$ is the reward collected in time slot t under the decision policy π with the belief state vector $\Omega(t)$. The initial belief state vector is $\Omega(1)$.

The optimal solution of Equation (8) can be obtained recursively via dynamic programming. The value function over T time slots is defined as:

$$V_T\big(\Omega(T)\big) = max_{\substack{A(T)\subseteq\{1,2,\ldots,N\}\\|A(T)|=K}} E\big[R_{A(T)}(\Omega(T))\big] = max_{\substack{A(T)\subseteq\{1,2,\ldots,N\}\\|A(T)|=K}} E\big[\sum_{j\in A(T)} s_j(T)\big] \tag{9}$$

$$V_t\big(\Omega(t)\big) = max_{\substack{A(t)\subseteq\{1,2,\ldots,N\}\\|A(t)|=K}} E\big[R_{A(t)}\big(\Omega(t)\big) + \sum_{\varepsilon\subseteq A(t)} \prod_{j\in\varepsilon} \omega_j(t) \prod_{j\in A(t)\setminus\varepsilon}(1-\omega_j(t))V(\Omega(t+1))\big] \tag{10}$$

Note that $V_t(\Omega(t))$ represents the maximum expected remaining reward that can be accrued from time slots t ($1 \le t \le T$) to T. ε denotes the collection of backup VMs in state "1" under policy $A(t)$ and thereby, $A(t) \setminus \varepsilon$ denotes the collection of backup VMs in state "0". The belief state vector $\Omega(t+1)$ is updated according to the rule in Equation (7).

Considering the huge number of backup VMs in the cloud environment, the state space of backup VMs becomes enormous, and the optimal solution of the value function is usually computationally intractable. Hence, we propose a heuristic greedy algorithm which omits the impact of the current action on the future reward, and solely focuses on maximizing the expected immediate reward. Thus, we can greatly reduce the computational complexity from an n-dimensional optimization problem to n one-dimensional problems. With the known failure and restoration models of primary VMs, the objective of the heuristic time sharing policy is given by:

$$A^*(\Omega(t)) = arg\max_{\substack{A(t)\subseteq\{1,2,\ldots,N\}\\|A(t)|=K}} \sum_{j\in A(t)} \omega_j(t) \tag{11}$$

That is, the only thing to do is to determine the decision action $A(t)$ that maximizes the immediate reward. In detail, sort the elements of $\Omega(t)$ in descending order in time slot t, and select the backup VMs with the first K largest values in $\Omega(t)$. Before starting the heuristic time sharing policy, the state-transition matrices of the backup VMs and the initial belief state vector $\Omega(1)$ are determined according to the failure and restoration models of the primary VMs. A complete description of the heuristic time sharing policy is given in **Fig. 5**.

For performance evaluation of the heuristic time sharing policy, we propose four metrics: backup assurance rate, idle time utilization of backup VMs, utilization of backup VMs and policy regret. The service availability is guaranteed if there exists a backup VM to take over the work for each failed secondary VM; otherwise, the service availability decreases due to insufficient backup VMs for failed service VMs. In T time slots, the backup assurance rate under policy π is defined as:

$$A_\pi^T = min_{t\in\{1,2,\dots,T\}}(1, \frac{R_\pi(S(t))}{f_t}) \tag{12}$$

where f_t denotes the number of failed secondary VMs in time slot t.

Each backup VM serves its dedicated primary VM and can be shared by secondary VMs during its idle time using the heuristic time sharing policy. The idle time utilization of backup VMs using policy π in T time slots is given by:

$$U_\pi^j = \frac{\sum_{t=1}^T Is_\pi^j(t)}{\sum_{t=1}^T s_p^j(t)}, U_\pi = \frac{\sum_{j=1}^N U_\pi^j}{N} \tag{13}$$

where U_π^j denotes the idle time utilization of backup VM j, U_π denotes the average idle time utilization of all the backup VMs, $s_p^j(t)$ is the state of primary VM j in time slot t, and $Is_\pi^j(t)$ represents the indicator function:

$$Is_\pi^j(t) = \begin{cases} 1 & if j \in A(t), s_j(t) = 1, o_\pi^j(t) = 1 \\ 0 & if j \in A(t), s_j(t) = 0 \; or \; if j \notin A(t) \end{cases} \tag{14}$$

in which $o_\pi^j(t) = 1$ indicates that backup VM j is occupied by a failed secondary VM in time slot t using policy π.

The utilization of backup VMs using policy π in T time slots can then be given by:

$$\mathrm{US}_\pi^j = \frac{\sum_{t=1}^T (\mathrm{Is}_\pi^j(t) + (1 - s_p^j(t)))}{T}, \mathrm{US}_\pi = \frac{\sum_{j=1}^N \mathrm{US}_\pi^j}{N} \tag{15}$$

where US_π^j denotes the utilization of backup VM j, and US_π denotes the average utilization of all the backup VMs.

Regret of the heuristic time sharing policy can be regarded as the lost reward value compared with the optimal policy when the states of all the backup VMs can be observed, which is defined as:

$$regret_\pi^T = \sum_{t=1}^T (\sum_{j=1}^K \mu_j(t) - R_\pi(S(t))) \tag{16}$$

where $\mu_j(t)$ denotes the jth largest element of $S(t)$.

Algorithm: The Heuristic Time Sharing Policy

Input: the state-transition matrices of backup VMs, the initial belief state

$\Omega(1) = \{\omega_1(1), \dots, \omega_N(1)\}$ where $\omega_j(1) = p_{01}^j/(p_{01}^j + p_{11}^j), j \in \{1,2, \dots, N\}$.

Output: the decision policy $A(t)$ containing K VMs to be observed in time slot t.

Begin

1. while $j \le N$ do

2. update the belief state of backup VM j

$$\omega_j(t) = \begin{cases} p_{01}^j & if\ j \in A(t-1), s_j(t-1) = 0 \\ p_{11}^j & if\ j \in A(t-1), s_j(t-1) = 1 \\ \omega_j(t-1)p_{11}^j + (1-\omega_j(t-1))p_{01}^j & if\ j \notin A(t-1) \end{cases}$$

where $s_j(t-1)$ is the state of backup VM j in time slot $t-1$.

3. end while

4. sort $\omega_j(t), j \in \{1,2, \dots, N\}$ in descending order

5. choose the K largest $\omega_j(t), j \in \{1,2, \dots, N\}$ and the corresponding $j's$ as the value of $A(t)$

End.

Fig. 5. The description of the heuristic time sharing policy

4.2 Optimality of the Heuristic Time Sharing Policy

1 Homogeneous case

Ahmad et al. [22] proved that the greedy policy is optimal under the condition $p_{11}^j > p_{01}^j$ for a RMAB problem with N bandits, in which the two-state ("1" or "0") of each bandit evolves as an independent and identically distributed Markov process, a player can select K bandits to play in each time slot, and a reward is obtained whenever the player plays a bandit in state "1". Based on [22], we have Theorem 1 as follows.

Theorem 1 The heuristic time sharing policy for backup VMs achieves optimality under the condition $\frac{1}{E(D_f^j)} + \frac{1}{E(D_r^j)} < 1$.

Proof: Consider $\frac{1}{E(D_f^j)} + \frac{1}{E(D_r^j)} < 1$, for $T_1^j = E(D_f^j)$ and $T_0^j = E(D_r^j)$, we then have $\frac{1}{T_1^j} + \frac{1}{T_0^j} < 1$, which leads to $\frac{T_1^j - 1}{T_1^j} > \frac{1}{T_0^j}$. Note that $p_{11}^j = \frac{T_1^j - 1}{T_1^j}$ and $p_{01}^j = \frac{1}{T_0^j}$, it thus follows that $p_{11}^j > p_{01}^j$, under which the greedy policy is optimal. So the proposed heuristic time sharing policy for the backup VMs achieves optimality under the condition $\frac{1}{E(D_f^j)} + \frac{1}{E(D_r^j)} <$ 1.

2 Heterogeneous case

Wang et al. [30] presented a generic study on the optimality of the greedy policy for a RMAB problem with N bandits, in which the two-state ("1" or "0") of each bandit evolves as independent and non-identically distributed Markov processes, a player selects K bandits to play in each time slot, and a certain amount of reward is obtained. They established the closed-form conditions under which the greedy policy is guaranteed to be optimal. Based on [30], we have Theorem 2 as follows.

Theorem 2 The heuristic time sharing policy for backup VMs achieves optimality under the condition $\sum_{i=1}^{T}\left(max_{j\in N}\left(\frac{E(D_f^j)-1}{E(D_f^j)}-\frac{1}{E(D_r^j)}\right)\right)^i<1$.

Proof: For the convenience of presentation, we sort the belief states of all the backup VMs in time slot t in descending order $\Omega(t)=[\omega_1(t),\dots,\omega_N(t)]$, such that $A(t)=\{1,\dots,K\}$ and let $\Omega_{A(t)}(t)\triangleq\{\omega_j(t),j\in A(t)\}=\{\omega_1(t),\dots,\omega_K(t)\}$. Let the immediate reward function $F\left(\Omega_{A(t)}(t)\right)=\sum_{j=1}^{K}\omega_j(t)$ denote the sum of the values of belief states of the K selected backup VMs in time slot t.

First, $\forall i,j\in A(t)$, we have

$$\begin{aligned}F\left(\omega_1(t),\dots,\omega_i(t),\dots,\omega_j(t),\dots,\omega_K(t)\right)&=\omega_1(t)+\cdots+\omega_i(t)+\cdots+\omega_j(t)+\cdots+\omega_K(t)\\&=\omega_1(t)+\cdots+\omega_j(t)+\cdots+\omega_i(t)+\cdots+\omega_K(t)\\&=F(\omega_1(t),\dots,\omega_j(t),\dots,\omega_i(t),\dots,\omega_K(t))\end{aligned}$$

Thus the function $F(\Omega_{A(t)}(t))$ is symmetrical.

Secondly, $\forall j\in A(t)$, we have

$$F\left(\omega_1(t),\dots,\omega_j'(t),\dots,\omega_K(t)\right)=\omega_1(t)+\cdots+\omega_j'(t)+\cdots+\omega_K(t)$$
$$F\left(\omega_1(t),\dots,\omega_j(t),\dots,\omega_K(t)\right)=\omega_1(t)+\cdots+\omega_j(t)+\cdots+\omega_K(t)$$

Consider $\omega_j'(t)>\omega_j(t)$, then $\omega_1(t)+\cdots+\omega_j'(t)+\cdots+\omega_K(t)>\omega_1(t)+\cdots+\omega_j(t)+\cdots+\omega_K(t)$, it thus follows that $F\left(\omega_1(t),\dots,\omega_j'(t),\dots,\omega_K(t)\right)>F(\omega_1(t),\dots,\omega_j(t),\dots,\omega_K(t))$.

Thus the function $F(\Omega_{A(t)}(t))$ is monotonic.

Thirdly, $\forall j\in A(t)$, we have

$$F\left(\omega_1(t),\dots,\omega_j(t),\dots,\omega_K(t)\right)=\omega_1(t)+\cdots+\omega_j(t)+\cdots+\omega_K(t)$$
$$=\omega_j(t)\left(\omega_1(t)+\cdots+\omega_{j-1}(t)+\omega_{j+1}(t)+\cdots+\omega_K(t)\right)+\omega_j(t)+\left(\omega_1(t)+\cdots+\omega_{j-1}(t)+\omega_{j+1}(t)+\cdots+\omega_K(t)\right)-\omega_j(t)\left(\omega_1(t)+\cdots+\omega_{j-1}(t)+\omega_{j+1}(t)+\cdots+\omega_K(t)\right)$$
$$=\omega_j(t)\left(\omega_1(t)+\cdots+\omega_{j-1}(t)+1+\omega_{j+1}(t)+\cdots+\omega_K(t)\right)+\left(1-\omega_j(t)\right)\left(\omega_1(t)+\cdots+\omega_{j-1}(t)+0+\omega_{j+1}(t)+\cdots+\omega_K(t)\right)$$
$$=\omega_j(t)F\left(\omega_1(t),\dots,1,\dots,\omega_K(t)\right)+(1-\omega_j(t))F(\omega_1(t),\dots,0,\dots,\omega_K(t))$$

Thus the function $F\left(\Omega_{A(t)}(t)\right)$ is decomposable.

So $F\left(\Omega_{A(t)}(t)\right)$ is regular because of its symmetry, monotonicity and decomposability

[30].

Fourthly, consider $\sum_{i=1}^{T}\left(max_{j\in N}\left(\frac{E(D_f^j)-1}{E(D_f^j)}-\frac{1}{E(D_r^j)}\right)\right)^i<1$, for $T_1^j=E(D_f^j)$ and $T_0^j=E(D_r^j)$, we then have $\sum_{i=1}^{T}\left(max_{j\in N}\left(\frac{T_1^j-1}{T_1^j}-\frac{1}{T_0^j}\right)\right)^i<1$. Note that $p_{11}^j=\frac{T_1^j-1}{T_1^j}$ and $p_{01}^j=\frac{1}{T_0^j}$, let $\omega_{-j}(t)\triangleq\{\omega_i(t),i\in A(t),i\neq j\}$, it thus follows that $\max_{j\in N,\omega_{-j}(t)\in[0,1]^{k-1}}\left\{F\left(1,\omega_{-j}(t)\right)-F\left(0,\omega_{-j}(t)\right)\right\}\sum_{i=1}^{T}\left(max_{j\in N}\left(p_{11}^j-p_{01}^j\right)\right)^i<\min_{j\in N,\omega_{-j}(t)\in[0,1]^{k-1}}\left\{F\left(1,\omega_{-j}(t)\right)-F\left(0,\omega_{-j}(t)\right)\right\}$ which is the condition for the greedy policy to achieve optimality when $F\left(\Omega_{A(t)}(t)\right)$ is regular [30]. So the proposed heuristic time sharing policy for the backup VMs achieves optimality under the condition $\sum_{i=1}^{T}\left(max_{j\in N}\left(\frac{E(D_f^j)-1}{E(D_f^j)}-\frac{1}{E(D_r^j)}\right)\right)^i<1$.

4.3 Complexity Analysis

In each time slot t, the heuristic policy sorts the reward values of all the backup VMs and updates the belief state of each backup VM. When adopting the quick sort algorithm with N backup VMs, the time complexity of sorting the reward values is $O(N\,log\,N)$ and the time complexity of updating the belief state is $O(N)$. The overall time complexity is $O(N\,log\,N)$. Thus, in T time slots, the time complexity is $O(T*N\,log\,N)$. Compared with the exponential complexity imposed by dynamic programming methods, the time complexity is significantly reduced using the heuristic time sharing policy.

5. Experimental Evaluation

In this section, the evaluation of the heuristic time sharing policy is given with numeric simulation experiment and a prototype system deployed in a small-scale cloud platform. The simulation experiment simulates the "idle-busy" feature of backup VMs in a large-scale cloud system and verifies the performance of the heuristic algorithm. The experiment in the implemented prototype system can verify the validity in a real system.

5.1 Simulation Experiment

At the beginning of the numerical simulation, we generate the "normal-failure" data of each primary VM according to the failure and restoration models using Matlab. The "normal" random number represents the duration between primary VM failures, while the "failure" random number represents the primary VM restoration time. Then, we discretize the random numbers into a binary sequence consisting of "1" and "0", which denotes the

states of a primary VM in each time slot. **Fig. 6** shows the generated random numbers and discretization process when the failure model follows Poisson distribution with parameter λ=4, and the restoration model follows a Lognormal distribution with parameters μ=1, δ=0.2. The "normal-failure" data of each primary VM corresponds to the "idle-busy' data of its dedicated backup VM. Thus the binary sequence also denotes the states of a backup VM.

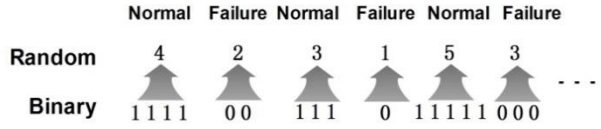
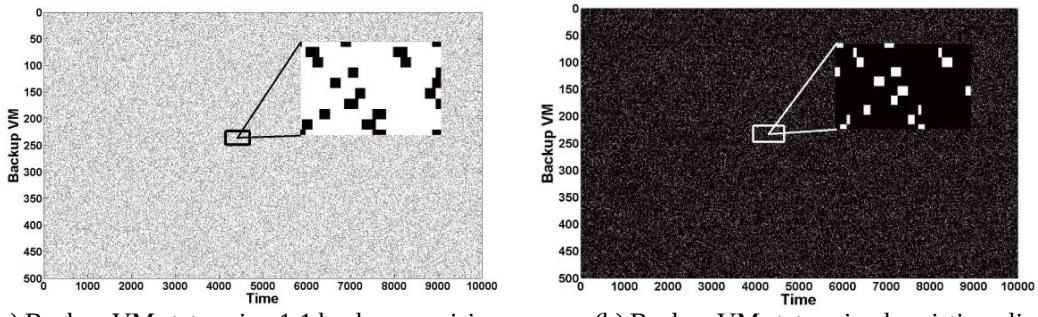


Fig. 6. Random numbers and discretization process

In the experiment, we simulate N (varying from 50 to 500) primary VMs and their N dedicated backup VMs using different failure and restoration models. The length of decision time T is 10000. We use a computer equipped with Intel i7-3770 CPU (3.40GHz), 4GB memory and 64-bit Windows 7 to conduct the simulation experiment. The heuristic time sharing policy is implemented in Java.

1 Utilization of backup VMs

We employ Poisson distribution failure model with λ=19 and Lognormal distribution restoration model with μ=1, δ=0.2. The per-backup VM states in 10000 time slots are shown in the form of heatmap (idle state in white and busy state in black) in **Fig. 7**, where the VM states using the traditional 1:1 backup provision are presented in **Fig. 7(a)**, and the VM states using the heuristic time sharing with K=400 and 7000 secondary VMs are presented in **Fig. 7(b)**. The magnified images clearly show the backup VM states before time sharing and after scheduling with the heuristic time sharing policy. In **Fig. 7(a)**, black



(a) Backup VM state using 1:1 backup provision (b) Backup VM state using heuristic policy

Fig. 7. States of 500 backup VMs in 10000 time slots

dots are scattered due to sparse failures of primary VMs, which shows the utilization of backup VMs is low before scheduling. In **Fig. 7(b)**, white regions are significantly reduced meaning that most of the idle backup VMs are shared by secondary VMs and hence the utilization of backup VMs is increased.

Fig. 8 shows the idle time utilization of backup VMs with different failure and restoration models, where the backup VMs are shared by 7000 secondary VMs with K=400. The failure rate of the primary VMs is fixed at 0.1 and Lognormal distribution is chosen as the restoration model with μ=1, δ=0.2 in all the cases. In the heterogeneous case as shown in **Fig. 8(a)**, the failure model is randomly selected from Poisson distribution with λ=19 and Weibull distribution with α=19, β=1, and it can be seen that the average idle time utilization of backup VMs marked with a dotted line is up to 87%. In the homogeneous cases, Poisson distribution is chosen as the failure model with λ=19 as shown in **Fig. 8(b)**, and Weibull distribution is chosen as the failure model with α=19, β=1 as shown in **Fig. 8(c)**. It can be found from **Fig. 8(b)-(c)** that the average values of the idle time utilization are all about 88% in the homogeneous cases. It can also be seen that there is nearly no difference in the idle time utilization between the homogeneous and heterogeneous cases, which indicates that the idle time utilization is independent on failure and restoration models when the failure rate is fixed.

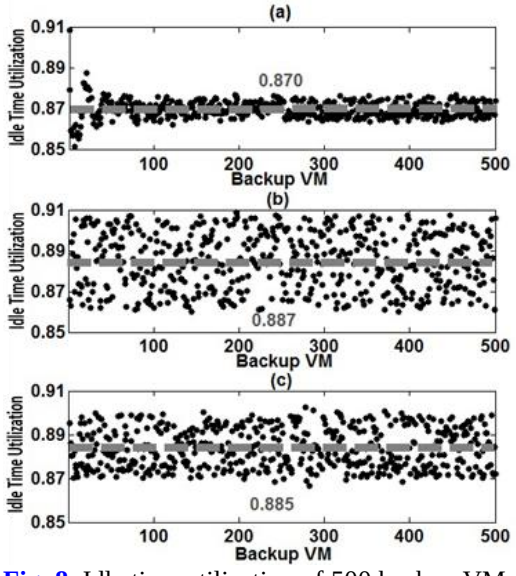


Fig. 8. Idle time utilization of 500 backup VMs over 10000 time slots with K=400. (a) Heterogeneous case; (b) Poisson distributed failure model and Lognormal distributed restoration model; (c) Weibull distributed failure model and Lognormal distributed restoration model

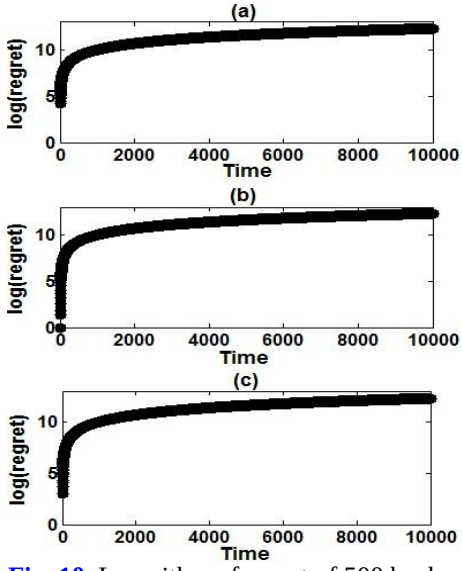


Fig. 10. Logarithm of regret of 500 backup VMs over 10000 time slots with K=400. (a) Heterogeneous case; (b) Poisson distributed failure model and Lognormal distributed restoration model; (c) Weibull distributed failure model and Lognormal distributed restoration model

To demonstrate the effectiveness of our approach, we compare it with the traditional 1:1 provision, optimal selection and random selection policies, and the corresponding average

utilizations are shown in **Fig. 9**, in which the models and parameters are the same as in **Fig. 8(c)**. In each time slot, the optimal selection policy can find out all the idle VMs with a post-analysis, while the random selection policy selects K backup VMs randomly. It can be found from the figure that the average utilization of backup VMs is the lowest using the traditional 1:1 backup provision, because it provides each service VM with a dedicated backup VM and no backup VM is shared. It can also be observed clearly that the average utilization of backup VM using the heuristic time sharing policy reaches 0.897, very close to the utilization 0.900 obtained by the optimal selection policy, and much higher than the random selection policy, which reveals the advantage of the heuristic time sharing policy in the idle backup VM selection according to the one-step look-ahead reward.

2 Logarithm of regret

Fig. 10 shows the logarithm of regret versus time, in which the models and parameters are the same as in **Fig. 8**. It can be observed that either in the heterogeneous case in **Fig. 10(a)** or in the homogeneous cases in **Fig. 10(b)-(c)**, the logarithm of regret converges to a constant. This indicates that the heuristic policy has the logarithmic optimal property. Because the states of backup VMs are continuously detected with increasing decision time, the prediction accuracy for the unknown states of the backup VMs will be enhanced and the decision policy will approach the optimum. It can also be found that the logarithms of regrets are bounded by the same value when the failure rates are the same in both the homogeneous and heterogeneous cases.

3 Upper bound of secondary VMs

For guaranteeing system availability and fully using resources, the maximum number of secondary VMs, also called the upper bound of secondary VMs, for which backup VMs can provide 100% backup assurance rate, is of importance. **Fig. 11** shows the backup assurance rate versus number of secondary VMs at different failure rates, where the number of primary VMs is 500 and K is 400. Poisson distribution is chosen as the failure model, Lognormal distribution is chosen as the restoration model, and the upper bounds are marked with vertical dotted lines. The different failure rates of primary VMs are obtained by modifying the parameters of the failure and restoration models, and **Table 1** presents the failure rates and the corresponding model parameters used in **Fig. 11**. It can be seen that the upper bounds of secondary VMs are 7500, 7000, 6000 and 1000, respectively. So the ratios of backup VMs to service VMs are 1:16, 1:15, 1:13 and 1:3, respectively, using the heuristic time sharing policy. As the failure rate increases, the idle time of backup VMs reduces, hence the number of service VMs that the 500 backup VMs can serve decreases accordingly. From the point of saving backup resources, the number of backup VMs can be reduced by two-thirds even at a failure rate of 0.2 as compared with the traditional 1:1 backup provision. Thus, the heuristic time sharing policy can achieve efficient reduction of virtual resources and extend the traditional 1:1 backup provision to 1:M ($M \gg 1$) between the backup VMs and the service VMs with the guaranteed backup assurance rate.

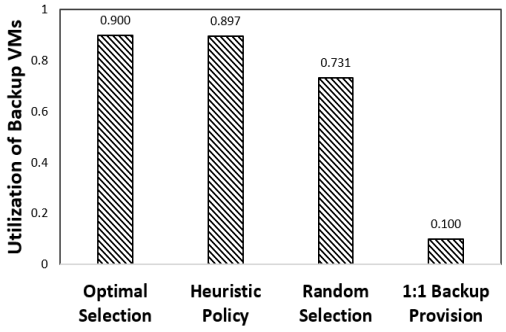


Fig. 9. Average utilization of backup VMs under different policies

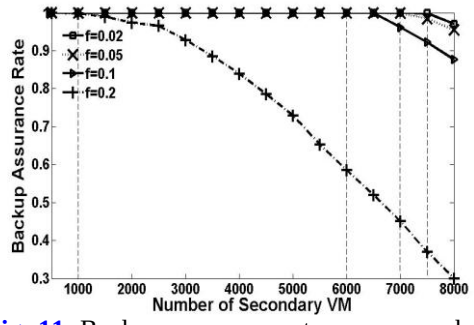


Fig. 11. Backup assurance rate versus number of secondary VMs at different failure rates

Table 1. Failure rates versus model parameters

Failure rate	Poisson	Lognormal
0.02	λ=135	μ=1,δ=0.2
0.05	λ=49	μ=1,δ=0.2
0.1	λ=19	μ=1,δ=0.2
0.2	λ=9	μ=1,δ=0.2

Fig. 12 shows the backup assurance rate versus the number of secondary VMs with different failure and restoration models, where the models and parameters are the same as in **Fig. 8**. It can be found that the upper bounds of the secondary VMs are all about 6000 in the three cases. So it indicates that the upper bound is nearly independent on failure and restoration models when the failure rate is fixed.

According to the provision, secondary VMs can only use idle backup VMs but cannot preemptively use busy backup VMs. Therefore, secondary VMs have no stringent availability guarantee. Extreme cases may happen, in which a majority of VMs fail, causing terrible availability. However the extreme cases are rare. Even in the very bad case where the failure rate is unrealistically as high as 0.9, indicating that 90% VMs fail in each time slot, the system availability is 0.598 in the 1:2 backup provision. Considering that primary VMs are provided with dedicated backup VMs, the availability of all primary VMs can be guaranteed and the low system availability is due to lack of backup VMs for failed secondary VMs. When the number of backup VMs is fixed, more secondary VMs result in lower system availability. When the extreme cases occur, backup VMs can still provide backup services for a certain number of secondary VMs. If the required availability of secondary VMs is high, we can simply add the number of backup VMs to guarantee the system availability. An alternative method for counteracting the extreme cases is to set different priorities for secondary VMs based on their availability requirements. A higher availability requirement corresponds to a higher priority. In the rare extreme cases, the limited idle backup VMs are assigned to the secondary VMs with higher priorities, for which the heuristic time sharing policy can be easily modified. Thus, the system availability can be further improved.

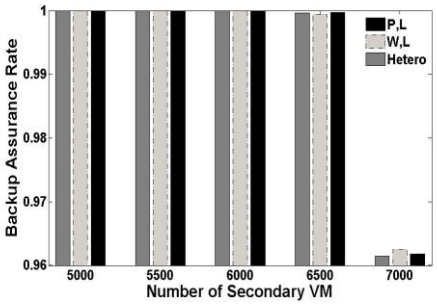


Fig. 12. Backup assurance rate versus number of secondary VMs with different models. P, L, and W denote Poisson, Weibull and Lognormal distributions, respectively

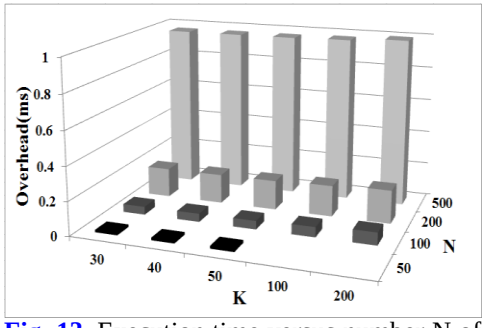


Fig. 13. Execution time versus number N of backup VMs and number K of selected backup VMs

4 Execution time evaluation

Fig. 13 shows the execution time of the heuristic time sharing policy versus the number N of backup VMs and the number K of selected backup VMs in the form of three-dimensional bar graph. We employ Poisson distributed failure model with λ=19 and Lognormal distributed restoration model with μ=1, δ=0.2. The execution time ranges from 0.01ms to 0.99ms. Because the execution time of the proposed policy is low enough, it can be employed for real-time decision. Another observation is that the execution time changes rarely when N is fixed, while the execution time increases sharply when N is increased, which is because the execution time is mainly spent on computing and sorting the reward values of backup VMs, as analyzed in Section 4.

5.2 Prototype System Verification

To verify the effectiveness of the heuristic time sharing policy in a real system, we design and implement a prototype system. The prototype system consists of three parts: State-transition Generation Module, Scheduling Decision Module and VM Monitoring Module. **Fig. 14** shows the architecture of the prototype system. State-transition Generation Module is deployed in Domain 0 of the physical machine hosting the primary VMs. This module monitors the primary VMs and generates the state-transition matrices of backup VMs, which consists of "0" and "1" depicting the two-states, namely: busy and idle, according to the VM failure and restoration models. Scheduling Decision Module is the core module of the prototype system, which is responsible for selecting a group of backup VMs for time sharing by secondary VMs. VM Monitoring Module is deployed in Domain 0 of the physical machine hosting the secondary VMs. This module monitors the secondary VMs and records the secondary VMs which are about to fail, then migrates such VMs to the backup VMs in state "1" according to the result of Scheduling Decision Module. As shown in **Fig. 14**, VM S1 and VM Sn are migrated to VM B2 and VM B3, respectively.

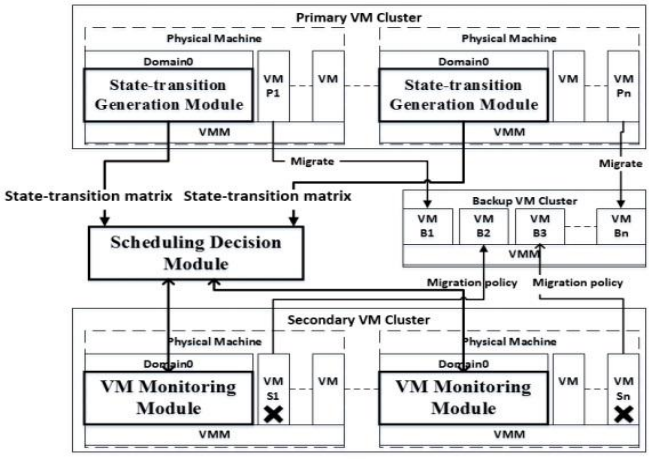


Fig. 14. Architecture of the prototype system

We deploy the prototype system in a small-scale cloud platform with 4 physical machines. Each physical machine is equipped with an 8-core Xeon processor (2.1GHz), 16GB memory and 64-bit Cent OS 6.2. One of the physical machines is used as the NFS server. There are 12 VM instances, which are divided into three parts: 4 primary VMs running on the second physical machine with identical failure and restoration models, their 4 dedicated backup VMs running on the third physical machine, and 4 secondary VMs running on the fourth physical machine. Each VM occupies 2 vcpu and 2GB memory. In order to simulate failures of the primary VMs, we generate the "normal-failure" data for each primary VM using Poisson distributed failure model with λ=19 and Lognormal distributed restoration model with μ=1, δ=0.2. When a primary VM fails, it is migrated to its dedicated backup VM. To simulate failures of the secondary VMs, 2 out of 4 secondary VMs are selected randomly to trigger failures and migration operations in each decision time slot. For the purpose of validating the effectiveness of the algorithm, failures and migration operations are recorded in Domain 0.

Fig. 15 shows the states of the backup VMs in 50 minutes before scheduling with the heuristic time sharing policy (idle state in white and busy state in black). The backup VMs are idle in most of the time, since they can only be occupied by the primary VMs with sparse failures. The utilizations of backup VM1, VM2, VM3 and VM4 in **Fig. 15** are 4%, 10%, 6% and 12% respectively. **Fig. 16** shows the states of the backup VMs in 50 minutes after scheduling (the idle state in white and the busy state in black). It can be seen that the black region is expanded, which implies that the backup VMs are shared by secondary VMs with the heuristic time sharing policy. The utilizations of backup VM1, VM2, VM3 and VM4 in **Fig. 16** are up to 52%, 34%, 80% and 64% respectively. The average utilization of the 4 backup VMs is 58% with the heuristic time sharing policy. The average idle time utilization of the 4 backup VMs is 54% using the proposed policy.

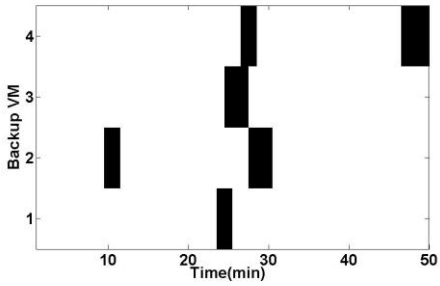


Fig. 15. Backup VM state before scheduling

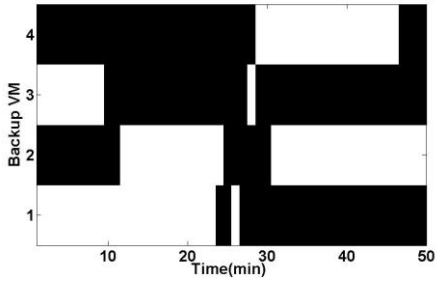


Fig. 16. Backup VM state after scheduling

6. Related Work

The over-provisioning of compute resources in the cloud to ensure high availability of services continues to be prohibitive in realizing high utilization. Improving resource utilization in modern cloud platforms has been identified as a critical design goal for reducing TCO for cloud providers [31]. To reduce TCO, the cloud owners take advantage of virtualization technique to flexibly isolate and share physical resources. Thus, seeking for efficient VM consolidation strategies has attracted extensive and in-depth discussions in academia and industry. Such strategies can be mainly classified into two categories: static VM consolidation and dynamic VM consolidation. Both categories of consolidation strategies formulate the resource allocation as a NP-hard multi-dimensional vector packing problem and establish a mapping between VMs and physical machines. The static VM consolidation strategies focus on the characteristics of VM workload and the usage of physical resources such as CPU and network bandwidth. Each VM is fixed in a physical machine according to the static VM consolidation strategy. This strategy is intent on improving resource utilization, reducing energy consumption and guaranteeing QoS in the cloud system. To ensure the high availability with k-resilient VMs, Bin [32] proposed a virtual resource allocation algorithm based on shadow VMs and the shadow VMs were deployed on specific physical machines to provide backup. To reduce the energy cost, Verma [33] analyzed the characteristics of real workload on servers and proposed a VM deployment algorithm based on the correlation between VMs. The static VM consolidation is commonly completed during the initialization of cloud infrastructure. As the workload and hardware equipment changes, the original deployment may not achieve the desired effect. Thus, dynamic VM deployment is adopted. The dynamic VM consolidation strategies adjust the VM deployment to adapt to the changing resources demand. Ahn [11] proposed a novel auto-scaling mechanism in order to dynamically adjust the number of VMs to handle deadline-critical real-time data in cloud computing infrastructures according to the volume of requested real-time tasks. Bobroff [34] proposed an algorithm which could predict the future resources of VMs with Auto-Regression model, and remap VMs to physical machines according to the resource requirement. Gong [35] dynamically

extracted patterns of resources requirement of different applications, and then performed dynamic VM deployment based on the extracted patterns.

The existing literatures address themselves to increase the virtual resource utilization in the production environment, while the utilization of backup virtual resources is out of consideration. Moreover, the traditional approaches cannot be used to optimize the backup resources, as they take into account the resource utilization instead of the failure probability of each VM. Wang [36] also aimed for optimizing backup resources but studied the configuration optimization of diesel generators and UPS to reduce the cost of backup power infrastructure. The most related works to ours are [21] and [22], where the authors formalize the open channel access as an RMAB problem. This paper is also an application of RMAB. However, the state transition between the VM states cannot be obtained directly, while in [21] and [22], it was a precondition. Therefore, we need to discretize the running state of each VM in each time slot in order to employ RMAB. To our knowledge, this paper presents the first study on the effective utilization of backup virtual resources with a bandit-based stochastic optimization model.

7. Conclusion

This paper proposes a heuristic time sharing policy based on the restless multi-armed bandit model, aiming to increase the backup resource utilization in the cloud system. The time sharing problem of backup VMs is formulated as a stochastic Markov process and the state of each backup VM evolves according to its state-transition matrix. We use the "belief state" of each backup VM as the scheduling indicator, thereby turning an n-dimensional optimization problem into n one-dimensional problems. Such decomposition reduces the computation complexity significantly. The proposed policy maximizes the utilization of the backup VMs by maximizing the reward value obtained in each time slot. In both the homogeneous and heterogeneous cases, optimality of the heuristic time sharing policy is also proved. Through the validation in simulation experiment and a prototype system, the proposed policy achieves the goal of extending the traditional 1:1 backup provision to 1:M ($M \gg 1$) between the backup VMs and the service VMs with the guaranteed system availability. Thus the utilization of backup VMs can be significantly enhanced.

References

[1] Amazon ec2, http://aws.amazon.com/ec2. Article (CrossRef Link)
[2] Windows azure, http://www.windowsazure.com. Article (CrossRef Link)
[3] Google compute engine, http://cloud.google.com/product%20s/compute-engine. Article (CrossRef Link)
[4] Vmware vcloud suite, http://www.vmware.com/products/%20vcloud. Article (CrossRef Link).
[5] Openstack cloud software, http://www.openstack.org. Article (CrossRef Link)
[6] Liang Yinglung, Yanyong Zhang, Morris Jette, Anand Sivasubramaniam and Ramendra Sahoo, "BlueGene/L failure analysis and prediction models," in *Proc. of IEEE International*

Conference on Dependable Systems and Networks, pp. 425-434, June 25-28, 2006. Aritcile (CrossRef Link)

[7] Zhang Yanyong, Mark S. Squillante, Anand Sivasubramaniam and Ramendra K. Sahoo, "Performance implications of failures in large-scale cluster scheduling," *Job Scheduling Strategies for Parallel Processing*, Springer Berlin Heidelberg, pp. 233-252, 2005. Article (CrossRef Link)

[8] Gagnaire Maurice, Felipe Diaz, Camille Coti, Christophe Cerin, Kazuhiko Shiozaki, Yingjie Xu, Pierre Delort et al, "Downtime statistics of current cloud solutions," *Tech. Rep of International Working Group on Cloud Computing Resiliency*, pp. 176-189, June, 2012. Article (CrossRef Link)

[9] Liu Jing, Jiantao Zhou and Rajkumar Buyya, "Software rejuvenation based fault Tolerance scheme for cloud applications," in *Proc. of 2015 IEEE 8th International Conference on Cloud Computing (CLOUD)*, pp. 1115-1118, June 27-July 2, 2015. Article (CrossRef Link)

[10] Bin Eyal, Ofer Biran, Odellia Boni, Erez Hadad, Eliot K. Kolodner, Yosef Moatti and Dean H. Lorenz, "Guaranteeing high availability goals for virtual machine placement," in *Proc. of 2011 31st International Conference on Distributed Computing Systems (ICDCS)*, pp. 700-709, June 20-24, 2011. Article (CrossRef Link)

[11] Yong W. Ahn, Albert M. K. Cheng, Jinsuk Baek, Minho Jo and Hsiao-Hwa Chen, "An auto-scaling mechanism for virtual resources to support mobile, pervasive, real-time healthcare applications in cloud computing," *IEEE Network*, vol. 27, no. 5, pp. 62-68, October, 2013. Article (CrossRef Link)

[12] Schroeder Bianca and Garth Gibson, "A large-scale study of failures in high-performance computing systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 337-350, October, 2010. Article (CrossRef Link)

[13] Chen Pengfei, Yong Qi, Pengfei Zheng, and Di Hou, "CauseInfer: automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems," in *Proc. of 2014 IEEE INFOCOM*, pp. 1887-1895, April 27-May 2, 2014. Article (CrossRef Link)

[14] Bruneo Dario, Salvatore Distefano, Federica Longo, Antonio Puliafito and Marco Scarpa, "Workload-based software rejuvenation in cloud systems," *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1072-1085, June, 2013. Article (CrossRef Link)

[15] Kishor S. Trivedi, Kalyanaraman Vaidyanathan, and Katerina Goševa-Popstojanova, "Modeling and analysis of software aging and rejuvenation," in *Proc. of 33rd Annual Simulation Symposium*, pp. 270-279, April, 2000. Article (CrossRef Link)

[16] Daniel R. Jeske, and Xuemei Zhang, "Some successful approaches to software reliability modeling in industry," *Journal of Systems and Software*, vol. 74, no. 1, pp. 85-99, January, 2005. Article (CrossRef Link)

[17] Okamura Hiroyuki, Tadashi Dohi and Shunji Osaki, "Software reliability growth models with normal failure time distributions," *Reliability Engineering & System Safety*, vol. 116, pp. 135-141, August, 2013. Article (CrossRef Link)

[18] Swapna S. Gokhale, "Software failure rate and reliability incorporating repair policies," in *Proc. of 10th International Symposium on Software Metrics*, pp. 394-404, September, 2004. Article (CrossRef Link)

[19] Robbins Herbert, "Some aspects of the sequential design of experiments," *Herbert Robbins Selected Papers*, pp. 169-177, 1985. Article (CrossRef Link)

[20] Whittle Peter, "Restless bandits: Activity allocation in a changing world," *Journal of applied probability*, pp. 287-298, 1988. Article (CrossRef Link)

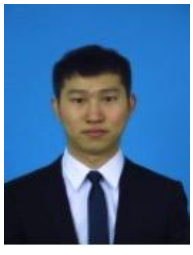
[21] Zhao Qing, Bhaskar Krishnamachari and Keqin Liu, "On myopic sensing for multi-channel

opportunistic access: structure, optimality, and performance," *IEEE Transactions on Wireless Communications*, vol. 7, no. 12, pp. 5431-5440, December, 2008. Article (CrossRef Link)
[22] Sahand H. A. Ahmad, and Mingyan Liu, "Multi-channel opportunistic access: A case of restless bandits with multiple plays," in *Proc. of 2009 Allerton 47th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1361-1368, September 30-October 2, 2009. Article (CrossRef Link)
[23] Cohen Kobi, Qing Zhao and Anna Scaglione, "Restless multi-armed bandits under time-varying activation constraints for dynamic spectrum access," in *Proc. of the 2014 48th IEEE Asilomar Conference on Signals, Systems and Computers*, pp. 1575-1578, November, 2014. Article (CrossRef Link)
[24] Bagheri Saeed and Anna Scaglione, "The restless multi-armed bandit formulation of the cognitive compressive sensing problem, " *IEEE Transactions on Signal Processing*, vol. 63, no. 5, pp. 1183-1198, March, 2015. Article (CrossRef Link)
[25] Sorensen Morten, "Learning by investing: evidence from venture capital," *AFA 2008 New Orleans Meetings Paper*, February, 2008. Article (CrossRef Link)
[26] Barbara F. L. Scala and Bill Moran, "Optimal target tracking with restless bandits," *Digital Signal Processing*, vol. 16, no. 5, pp. 479-487, September, 2006. Article (CrossRef Link)
[27] Alonso Javier and Kishor S. Trivedi, "Software rejuvenation and its application in distributed systems," *Quantitative Assessments of Distributed Systems: Methodologies and Techniques*, pp. 301-325, April, 2015. Article (CrossRef Link)
[28] Zheng Junjun, Hiroyuki Okamura and Tadashi Dohi, "Availability importance measures for virtualized system with live migration," *Applied Mathematics,* vol. 6, no. 2, pp. 359-372, February, 2015. Article (CrossRef Link)
[29] David S Touretzky, "advances in neural information processing systems," in *Proc. of the 1995 Mit Press Conference*, vol. 8, 1996. Article (CrossRef Link)
[30] Wang Kehao and Lin Chen. "On optimality of myopic policy for restless multi-armed bandit problem: An axiomatic approach," *IEEE Transactions on Signal Processing,* vol. 60, no. 1, pp. 300-309, January, 2012. Article (CrossRef Link)
[31] Luiz Andre Barroso, Jimmy Clidaras and Urs Holzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis Lectures on Computer Architecture*, vol. 8, no. 3, pp. 1-154, July, 2013. Article (CrossRef Link)
[32] Eyal Bin, Ofer Biran, Odellia Boni, Erez Hadad, Eliot K. Kolodner, Yosef Moatti and Dean H. Lorenz, "Guaranteeing high availability goals for virtual machine placement," in *Proc. of the 31st IEEE International Conference on Distributed Computing Systems*, pp. 700-709, June 20-24, 2011. Article (CrossRef Link)
[33] Verma Akshat, Gargi Dasgupta, Tapan K. Nayak, Pradipta De, and Ravi Kothari, "Server workload analysis for power minimization using consolidation," in *Proc. of the 2009 conference on USENIX Annual Technical Conference*, pp. 28-28, May 21-25, 2009. Article (CrossRef Link)
[34] Bobroff Norman, Andrzej Kochut and Kirk Beaty, "Dynamic placement of virtual machines for managing sla violations," in *Proc. of IM'07 10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 119-128, May 21, 2007. Article (CrossRef Link)
[35] Gong Zhenhuan and Gu Xiaohui, "Pac: Pattern-driven application consolidation for efficient cloud computing," in *Proc. of 2010 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 24-33, August 17-19, 2010. Article (CrossRef Link)

[36] Wang Di, Sriram Govindan, Anand S. A. Kansal, Jie Liu and Badriddine Khessib, "Underprovisioning backup power infrastructure for datacenters," *ACM SIGPLAN Notices*, vol. 49, no. 4, pp. 177-192, April, 2014. Article (CrossRef Link)

Xinyi Li received B.Eng. in Computer Science from Xi'an Jiaotong University, China, in 2012. She is currently a joint Ph.D. candidate in the Department of Computer Science and Technology, Xi'an Jiaotong University, China and the Department of Computer Science, North Carolina State University, USA. Her research interests include software reliability, maintenance scheduling and Cloud computing.

Qi Yong received Ph.D. degree from Xi'an Jiaotong University, China. He is currently a full professor in the Department of Computer Science and Technology at Xi'an Jiaotong University, China. His research interests include operating systems, distributed systems, and cloud computing. He has led more than 10 research projects including National 863 High-Tch. Program, NSFC projects, IBM and Microsoft Research projects, etc. He has been awarded 3 Province or Ministry Science Progress prizes, and published more than 70 papers.



Pengfei Chen received B.S. and Ph.D. degrees from Xi'an Jiaotong University in 2009 and 2016 respectively. His research interests include dependable computing, cloud computing, big data systems, and distributed computing. He has published more than 20 technical papers in several high-rank international conferences (e.g., IEEE INFOCOM, IEEE BigData) and journals (e.g., IEEE TR, IEEE TETC, IEEE TDSC). He is also a student member of CCF and a reviewer of multiple international journals (e.g., IEEE Transaction on Cybernetics). Mr. Chen is currently a research scientist of IBM Research-China.