Optimal Task Admission Control of Private Cloud Data Centers With Limited Resources

Wenlong Ni*, Yuhong Zhang†, and Wei Li†
*99 ZiYang Ave, JiangXi Normal University, NanChang, CHINA
†3100 Cleburne St, Texas Southern University, Houston, USA

***Abstract*—In this paper, we study a private data center (PDC) serving both priority and non-priority tasks with limited resources. To make full use of the DC resources, e.g. virtual machines (VM), any task request in a non-priority task with a large computing requirement may exhaust all available VMs that is not currently used by the priority task. While receiving rewards for serving non-priority tasks, there is also the cost of holding and processing the tasks in the system. By balancing rewards and costs, our goal is to decide whether to accept or reject non-priority tasks. The first validation of optimization policies for embracing non-priority tasks is control limit policies by translating the problem into stochastic dynamic planning. The innovation of this paper is the development of DC related stochastic dynamic programming and the study of how to get clear optimization control limit policy results. These developments and studies are of great practical value for the theoretical analysis and design of various DC models with optimized reward and system utilization.**

***Index Terms*—Cloud Data Center, Cognitive Network, Optimal strategy, Cost Efficiency, bandwidth Allocation.**

I. Introduction

Cloud computing (CC) [1]–[6] is a special computing model that enables dynamic and scalable virtualization resources to be served over the Internet without requiring users to understand the technical details behind it. The CC describes the basic information services infrastructure, including networking, computing, storage, and software such as operating systems, application platforms, and network services. CC uses the concept of a "cloud" to emphasize the use of these resources rather than their implementation details. The rapid development of technology has led to a significant increase in the demand for data centers (DC).

As shown in Fig.1, virtualization technology enables the execution of multiple virtual machines (VM) on a single physical machine (PM), thereby supporting multiple client operating systems. The allocation of physical system resources, such as processors, memory, hard disks, and networks, can be controlled and apportioned to VMs in a precise manner. Each VM is completely isolated from the others, ensuring that if one or several VMs experience a crash, the other VMs will not be impacted, and data will not be leaked between them. When a user sends a service request to the DC, one or more VMs are assigned to handle it. The handling of tasks can be prioritized based on user privileges, task urgency, latency, workload, and other factors. The primary concern of any cloud service provider (CSP) is to ensure quality of service (QoS) under service level agreement (SLA) constraints while achieving maximum profitability [7]–[9]. In addition, a novel DC model of cognitive abilities with online tasks (priority) were investigated relative to offline batch tasks (non-priority) [10], which uses as much VM resources as available to serve both online and offline tasks and the number of VMs can vary time to time. Two semi-Markov decision process (SMDP)-based coordinated VM allocation methods were also proposed to balance the tradeoff between the high cost of providing services by the remote cloud and the limited computing capacity of the local fog [11].

A private cloud DC (PDC) provides enterprises with cloud-based IT management, increasing data sharing capabilities between business departments, improving resource utilization rates, reducing operational costs, and enabling automated operation and management compliance of systems. A PDC is usually with a limited number of hard wares. Based on the fact that the DC utilization rate is low [12] and the scientific computing demand is large [5], this paper proposes a new resource allocation scheme to use the VM resources in the PDC. The scheme regards the daily work of the PDC as a priority and the tasks that require a lot of computation as a non-priority. Without interfering with the routine work of the PDC, some features of the program include (1) there are two types of tasks: priority and non-priority tasks in the system; more specifically non-priority tasks are scientific computing tasks which need a lot of VMs to process and are willing to pay a price for the computing resources [7], [13]; (2) Priority tasks are time-sensitive tasks which need service as soon as possible; Non-priority tasks are computing tasks that can be processed in a parallel way; (3) Instead of waking/sleeping/moving VMs, it is better to make full usage of VM resources. To do this, all VMs that do not serve the priority tasks will be used to handle non-priority tasks.; (4) VMs that serve non-priority tasks can be preempted by new priority task; (5) If all the VMs are busy serving priority tasks, a new arrival priority task will be queued for future service; similarly for an arrival of non-priority task, it will also be queued if all VMs are busy.

In our recent work [14], we discuss the general case of the

This material is partially based upon work supported by the NSF under Grant No. 2302469 and 2318662, and by NASA under award number 80NSSC22KM0052 to Wei Li; and by JiangXi Education Department under Grant No. GJJ191688 to Wenlong Ni.

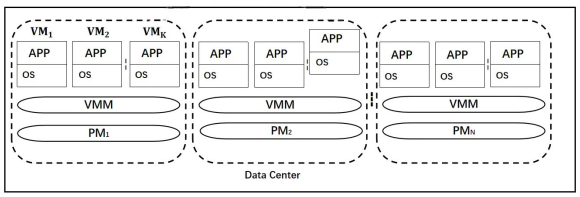

Fig. 1. PMs and VMs in a DC.

absence of cognitive function where the DC has the ability to determine the cost of utilizing the resource, while the cloud service users have the willingness to bear the resource cost. One of the major differences between the model in this published work and the current model in the submission is on the way of handling the process priority of different types of tasks in the system. Furthermore, the focus of our current paper is on optimizing PDC revenue by providing services for non-priority tasks. Our major contributions in this paper include:

1) The reward of serving non-priority tasks and the holding costs related to keeping both types of tasks in the system are studied. To achieve the optimal policy for determining when to admit or reject non-priority tasks, a stochastic dynamic programming model known as Continuous Time Markov Decision Process (CTMDP) [15], [16] is established to maximize the total expected discounted reward of the CSP.
2) To optimize the utilization of VM resources in the DC and maximize benefits, all unused VMs not occupied by priority tasks will be allocated to non-priority tasks. These non-priority tasks are computationally intensive and can be processed concurrently using multiple VMs. This research builds upon our previous work [14], extending it to address the added complexity of treating tasks equally in a regular DC and prioritizing tasks in a cloud DC.
3) In the proposed model, there is a buffer for both priority and non-priority tasks. The non-priority tasks may be preempted by incoming priority tasks. Through a systematic probability analysis, it has been verified that the optimal policy for admitting or rejecting non-priority tasks is a state-related control limit (threshold) policy.

The remainder of this paper is organized as follows. In Section II, the system model of the DC is introduced. The structure of optimal policy to maximize total expected discount reward, the verification process that it is a control limit policy and an boundary analysis of the threshold value is described in Section III. In Section IV, we present a numerical analysis with table and diagram that validate the theoretical results. Finally, Section V offers concluding remarks.

II. Model Description and Analysis

CTMDP can be used to model a dynamic stochastic process. In this section, we first introduce the system model of a DC with the assumptions of all needed parameters; then we show the establishment of important components in the corresponding CTMDP model.

A. The System Model

The detailed assumptions for the DC are given as follows:

1) There is a number of VMs, denoted by C, in the PDC. There are two types of tasks in the system, the priority task is simplified as type-1 (T_1) task and non-priority task as type-2 (T_2) task. To save energy, the idle VMs can go to sleep or wake up if needed. The number of VMs in a DC serving both types of the tasks is C, which changes dynamiclly depending on the number of tasks in the system.
2) T_1 is a priority (time-sensitive) task and needs a number of VMs for service; T_2 is a non-priority (parallel computing) task and can be processed with any number of VMs. For simplicity in this paper we assume the DC will provide a fixed number say b (here b is a given non-zero positive integer) VMs for each incoming T_1 task.
3) The arriving process for tasks T_1 and T_2 are Poisson processes [17] with rates λ_1 and λ_2, respectively. The task processing time for these tasks in one VM follows a negative exponential distribution with rates μ_1 and μ_2, respectively. If a $T_1(T_2)$ task is processed with $x, x \geq 1$ VMs, the service rate would be $x\mu_1(x\mu_2)$ for that task.

The processing rules of T_1 and T_2 tasks is listed as below:

1) When a T_2 task comes to the system, let $n_1(n_2)$ be the number of $T_1(T_2)$ tasks currently in the system; the pseudo code of what actions CSP will take is listed in the following algorithm 1:

Algorithm 1: Pseudo code for T_2 arrival processing rules

```
Parameters: $C, b$;
Initialization: $N_1 = \frac{C}{b}, n_1, n_2$;
if $n_1 < N_1$ and $n_2 = 0$ then
    Process the new $T_2$ task using $C - bn_1$ VMs
    $n_2 = n_2 + 1$
else
    if ACCEPT then
        Save the $T_2$ task in the buffer
        $n_2 = n_2 + 1$
    else
        $T_2$ task is Rejected and will leave the system
```

2) When a T_1 task comes to the system, let n_1 be the number of T_1 tasks currently in the system; the pseudo code of what actions CSP will take is listed in the following algorithm 2 :

The system is primarily designed to handle T_1 tasks, with $C = bN_1$ where N_1 represents a positive integer. Since the T_1 task is always accepted, our focus is on the admission control of the T_2 task. Serving a T_2 task results in a reward of R

Algorithm 2: Pseudo code for T_1 arrival processing rules

```
Parameters: $C, b$;
Initialization: $N_1 = \frac{C}{b}, n_1$;
if $n_1 < N_1$ then
    Process $T_1$ task with $b$ VMs
else
    Save the $T_1$ task in the buffer
$n_1 = n_1 + 1$
```

units being awarded to the CSP. However, if a VM serving a T_2 task is preempted by a higher priority T_1 task, there is a cost associated with that interruption, labeled as $r(r \geq 0)$. To maintain tasks within the system, the CSP must pay a holding price at a rate $f(n_1, n_2)$ to manage the VMs in the DCs when there are n_1 T_1 tasks and n_2 T_2 tasks in the system. It should be noted that some notation used above, including T_1, T_2, λ_1, λ_2, μ_1 and μ_2, follow the conventions established in [14] to facilitate ease of connection with our recent works in this area. All major given parameters are summarized in Table I for reference.

TABLE I
A LIST OF MAJOR GIVEN PARAMETERS

C	Number of VMs in the DC
α	Discount factor
λ_i	Arrival rate of T_i $(i = 1, 2)$
μ_i	Service rate of T_i $(i = 1, 2)$
b	Number of VMs needed to process a T_1 task
r	Cost of a VM serving T_2 interrupted by a T_1 task
R	Reward of serving a T_2
$f(n_1, n_2)$	Holding cost rate at state (n_1, n_2)

B. CTMDP Model and its Components Analysis

Based on the system assumptions, we can now establish a following CTMDP model:

1) State Space: In the CTMDP model, our focus is solely on the decision state, which comprises both traditional states and events at the time of decision-making. The first component of the decision state, known as the traditional state, is defined as the number of ongoing task types in the data center. This is expressed as $S = \{s : s = (n_1, n_2), n_1 \geq 0, n_2 \geq 0\}$, where n_1 and n_2 are two non-negative integers, representing the number of T_1 and T_2 tasks, respectively. C signifies the number of virtual machines. The second component of the decision state, the event, is represented as e. The event space is defined by $e \in E = \{D_1, D_2, A_1, A_2\}$, where D_i $(i = 1, 2)$ indicates the completion and departure of T_i $(i = 1, 2)$, and A_i $(i = 1, 2)$ signifies the arrival of a T_i $(i = 1, 2)$ task.
   Therefore, a decision-making state can be formulated as $\hat{s} = \langle s, e\rangle = \langle (n_1, n_2), e\rangle$. The state space is the set of all the available decision-making states, represented as
$$\hat{S} = S \times E = \{\hat{s} | \hat{s} = \langle (n_1, n_2), e\rangle\}.$$

2) Action Space: The controller will not make any decisions when the service is complete. We define a_C as an fictitious action of a service completion (departure), where $e \in \{D_1, D_2\}$ is any departure event. Therefore we have
$$A_{\langle (n_1,n_2),D_i\rangle} = \{a_C\}, n_i > 0, (i = 1, 2).$$
   When the priority task T_1 arrives, the only action is to admit (accept) it, meanwhile we can either admit or reject the arrival of non-priority task T_2. If we define a_A as an admission action and a_R as a rejected request action, then:
$$A_{\langle (n_1,n_2),A_2\rangle} = \{a_R, a_A\};$$
$$A_{\langle (n_1,n_2),A_1\rangle} = \{a_A\}.$$
   Thus, the action space is the set of all possible actions, defined as follows:
$$A = \{a_A, a_R, a_C\}.$$

3) Decision Epoch: The decision time point refers to the moment when a task reaches or departs from the system, or when an event occurs. Based on our hypothesis, the time duration between these decision points follows an exponential distribution, with a rate parameter $\beta(\hat{s}, a)$. The specific formula is given by:
$$F(t|\hat{s}, a) = 1 - e^{-\beta(\hat{s},a)t}, t \geq 0.$$
   Moreover, $\hat{s} = \langle((n_1, n_2)), e\rangle$. When the number of T_1 tasks is n_1, the number of T_2 tasks is n_2 , $V_1(n_1)$ indicates the number of VMs occupied by the T_1 task, $V_2(n_1, n_2)$ indicates the number of VMs occupied by T_2 tasks, $V_3(n_1, n_2)$ represents the number of VMs serving T_2 tasks that will be preempted if a T_1 task is admitted. These definitions are sometimes simplified as V_1, V_2 and V_3 in this paper.
$$V_1(n_1) = \begin{cases} C, & n_1 \geq N_1, \\ bn_1, & n_1 < N_1, \end{cases}$$
$$V_2(n_1, n_2) = \begin{cases} C - V_1(n_1), & n_2 > 0, \\ 0, & n_2 = 0, \end{cases}$$
$$V_3(n_1, n_2) = \begin{cases} b, & n_1 < N_1, n_2 > 0, \\ 0, & otherwise. \end{cases}$$
   Denote by $s = (n_1, n_2)$ and $\beta_0(s) = \lambda_1 + \lambda_2 + V_1\mu_1 + V_2\mu_2$, which means that the average time, the system changes from the state s to any other state, is $1/\beta_0(s)$. Specifically, we will have, $\beta_0(s) =$
$$\begin{cases} C\mu_1 + \lambda_1 + \lambda_2, & n_1 \geq N_1, \\ bn_1\mu_1 + \lambda_1 + \lambda_2, & n_1 < N_1, n_2 = 0, \\ bn_1\mu_1 + (C - bn_1)\mu_2 + \lambda_1 + \lambda_2, & n_1 < N_1, n_2 > 0, \end{cases}$$
   Given the definition of $\beta(\hat{s}, a)$ and $\beta_0(s)$, if s_n denotes the next state after taking action a from the current decision-making state $\hat{s}$, it is evident that
$$\beta(\hat{s}, a) = \beta_0(s_n).$$

4) Transition Probability: Given the decision-making state $\hat{s}$ and action a at the current decision epoch, let $q(j|\hat{s},a)$ represent the probability that the system occupies decision-making state j in the next epoch. For departure events, such as a departure event of D_1 when $(n_1 > 0)$, with $(\hat{s},a) = (\langle(n_1,n_2),D_1\rangle,a_C)$, if denote by $s_{d_1} = (n_1-1,n_2)$, then we have $q(j|\langle(n_1,n_2),D_1\rangle,a_C)$ as

$$\begin{cases} \lambda_1/\beta_0(s_{d_1}), & j=\langle s_{d_1},A_1\rangle,\\ V_1(n_1-1)\mu_1/\beta_0(s_{d_1}), & j=\langle s_{d_1},D_1\rangle,\\ \lambda_2/\beta_0(s_{d_1}), & j=\langle s_{d_1},A_2\rangle,\\ V_2(n_1-1,n_2)\mu_2/\beta_0(s_{d_1}), & j=\langle s_{d_1},D_2\rangle. \end{cases}$$

Similar equations can be derived for $(\hat{s},a) = (\langle(n_1,n_2),D_2\rangle,a_C)$ when $(n_2 > 0)$. If we denote the resulting state after the departure event by $s_{d_2} = (n_1,n_2-1)$, then we have $q(j|\langle(n_1,n_2),D_2\rangle,a_C)$ as

$$\begin{cases} \lambda_1/\beta_0(s_{d_2}), & j=\langle s_{d_2},A_1\rangle,\\ V_1(n_1)\mu_1/\beta_0(s_{d_2}), & j=\langle s_{d_2},D_1\rangle,\\ \lambda_2/\beta_0(s_{d_2}), & j=\langle s_{d_2},A_2\rangle,\\ V_2(n_1,n_2-1)\mu_2/\beta_0(s_{d_2}), & j=\langle s_{d_2},D_2\rangle. \end{cases}$$

For arrival events of A_1 and A_2, since admitting an incoming task immediately migrates the system state (by adding one task), if we denote the resulting state after the arrival events by $s_{a_1} = (n_1+1,n_2)$ and $s_{a_2} = (n_1,n_2+1)$, we obtain $q(j|\hat{s},a)$ as

$$\begin{cases} \lambda_1/\beta_0(s_{a_1}), & j=\langle s_{a_1},A_1\rangle,\\ V_1(n_1+1)\mu_1/\beta_0(s_{a_1}), & j=\langle s_{a_1},D_1\rangle,\\ \lambda_2/\beta_0(s_{a_1}), & j=\langle s_{a_1},A_2\rangle,\\ V_2(n_1+1,n_2)\mu_2/\beta_0(s_{a_1}), & j=\langle s_{a_1},D_2\rangle. \end{cases}$$

And, for action a_A related to A_2, we have $q(j|\langle(n_1,n_2),A_2\rangle,a_A)$ is

$$\begin{cases} \lambda_1/\beta_0(s_{a_2}), & j=\langle s_{a_2},A_1\rangle,\\ V_1(n_1)\mu_1/\beta_0(s_{a_2}), & j=\langle s_{a_2},D_1\rangle,\\ \lambda_2/\beta_0(s_{a_2}), & j=\langle s_{a_2},A_2\rangle,\\ V_2(n_1,n_2+1)\mu_2/\beta_0(s_{a_2}), & j=\langle s_{a_2},D_2\rangle. \end{cases}$$

Since action a_R rejects the arrival task, there is no change to the system, we have $q(j|\langle(n_1,n_2),A_2\rangle,a_R)$ is

$$\begin{cases} \lambda_1/\beta_0(s), & j=\langle s,A_1\rangle,\\ V_1(n_1)\mu_1/\beta_0(s), & j=\langle s,D_1\rangle,\\ \lambda_2/\beta_0(s), & j=\langle s,A_2\rangle,\\ V_2(n_1,n_2)\mu_2/\beta_0(s), & j=\langle s,D_2\rangle. \end{cases}$$

5) Reward Function: The system will be rewarded based on system states and the corresponding actions. The reward function is determined by the reward (income) $k(\hat{s},a)$ received from users and the system cost at rate $c(\hat{s},a)$. The total expected discount reward between epochs satisfies

$$\begin{aligned} r(\hat{s},a) &= k(\hat{s},a) + c(\hat{s},a)E_{\hat{s}}^{a}\{\tau_1\}\\ &= k(\hat{s},a) + \frac{c(\hat{s},a)}{\alpha+\beta(\hat{s},a)}, \end{aligned} \tag{1}$$

where

$$k(\hat{s},a) = \begin{cases} 0, & e=A_2, a=a_R,\\ R, & e=A_2, a=a_A,\\ 0, & e=\{D_1,D_2\}, a=a_C,\\ -V_3 r, & e=A_1, a=a_A. \end{cases}$$

Upon acceptance of a T_2 task, the reward is received upon completion of the service, which is equivalent to engaging in the accepted action. Furthermore, $c(\hat{s},a)$ represents the holding cost rate at state (n_1,n_2) subsequent to the execution of action a. Let $f(s), s=(n_1,n_2)$ denote the cost rate when the system is in state s. Then, $c(\hat{s},a)$ can be expressed as

$$c(\hat{s},a) = \begin{cases} -f(n_1+1,n_2), & e=A_1,\\ -f(n_1,n_2+1), & e=A_2, a=a_A,\\ -f(n_1-1,n_2), & e=D_1, n_1>0,\\ -f(n_1,n_2-1), & e=D_2, n_2>0,\\ -f(n_1,n_2), & e=A_2, a=a_R. \end{cases}$$

At each decision epoch, the policy clearly specifies the decision rules based on the current state (the action to be taken). For each policy π, $v_\alpha^\pi(\hat{s})$ represents the sum of the expected infinite-horizon discounted reward when the process is in state $\hat{s}$, where α is the discount factor. Based on the five components of CTMDP model described above, if $\hat{s}_n$ represents the state at decision epoch n, t_n is the time point of decision epoch, a_n be the action to take at state $\hat{s}_n$, and $r(\hat{s}_n,a_n)$ for the reward obtained during the decision epoch (from time t_n to t_{n+1}) after action a_n is selected. Our goal is to find an optimal policy π that can bring the maximum total expected discounted reward $v_\alpha^\pi(\hat{s})$ for every initial state $\hat{s}$.

$$v_\alpha^\pi(\hat{s}) = \lim_{n\to\infty} E_{\hat{s}}\left[\int_0^{t_n} e^{-\alpha t} r(\hat{s}_n,a_n)dt\right]. \tag{2}$$

III. Optimal Stationary State-related Control Limit Policy

In a policy, if there is a threshold for receiving tasks, then the policy is called a control limit policy (or a threshold policy). In this study, we mainly focused on the acceptance of the T_2 task. When n_1 T_1 tasks exist, a threshold $T(n_1) \geq 0$ is set in the system. As long as the number of the T_2 task in the system is less than $T(n_1)$, the system will accept the newly arrived T_2 task, otherwise it will be rejected. This means that the decision rule for the T_2 task is as follows:

$$d(n_1,n_2,A_2) = \begin{cases} a_A, & n_2 \leq T(n_1),\\ a_R, & n_2 > T(n_1). \end{cases} \tag{3}$$

It is clear that the threshold policy provides a simple decision basis for the decision maker (CSP).

According to the above equation, for a departure event of D_1, it is straightforward to have

$$\begin{aligned} & v(\langle(n_1+1,n_2),D_1\rangle)\\ = & \frac{1}{\alpha+\beta_0(n_1,n_2)}[-f(n_1,n_2) + \lambda_1 v(\langle(n_1,n_2),A_1\rangle)\\ & +n_1\mu_1 v(\langle(n_1,n_2),D_1\rangle) + \lambda_2 v(\langle(n_1,n_2),A_2\rangle)\\ & +n_2\mu_2 v(\langle(n_1,n_2),D_2\rangle)]. \end{aligned} \tag{4}$$

When considering results similar to the above departure events, we can consider an arrival event for the T_2 task and produce the following results:

$$\begin{aligned} & v(\langle (n_1,n_2),A_2\rangle, a_A) \\ = \;& R + \frac{1}{\alpha+\beta_0(n_1,n_2+1)}\Big[-f(n_1,n_2+1) \\ & +\lambda_1 v(\langle (n_1,n_2+1),A_2\rangle) + n_1\mu_1 v(\langle (n_1,n_2+1),D_1\rangle) \\ & +\lambda_2 v(\langle (n_1,n_2+1),A_2\rangle) \\ & +(n_2+1)\mu_2 v(\langle (n_1,n_2+1),D_2\rangle)\Big], \end{aligned}$$

and

$$\begin{aligned} & v(\langle (n_1,n_2),A_2\rangle, a_R) \\ = \;& \frac{1}{\alpha+\beta_0(n_1,n_2)}\Big[-f(n_1,n_2)+\lambda_1 v(\langle (n_1,n_2),A_1\rangle) \\ & +n_2\mu_2 v(\langle (n_1,n_2),D_2\rangle) + \lambda_2 v(\langle (n_1,n_2),A_2\rangle) \\ & +n_1\mu_1 v(\langle (n_1,n_2),D_1\rangle)\Big]. \end{aligned}$$

For the T_1 tasks, as the system always accepts them, there is interruption cost if a VM serving T_2 task is preempted by T_1, we have Based on the above equations, it can be observed that the value of $v(\hat{s})$ is primarily influenced by the values of n_1 and n_2 after performing the corresponding actions for the incoming event. Therefore, we can introduce a new function $X(s)$ with $s = (n_1,n_2)$ defined as follows:

$$X(n_1,n_2) = v(\langle (n_1,n_2+1),D_2\rangle) = v(\langle (n_1+1,n_2),D_1\rangle)$$

After analyzing these equations, it can be verified that, considering only the accept/reject actions for T_2 arrivals, the following equation holds:

$$v(\langle (n_1,n_2),A_2\rangle) = \max\Big[R + X(n_1,n_2+1), X(n_1,n_2)\Big]$$

For T_1 tasks, as the system always accepts them, there may be interruption costs if a VM serving T_2 task is preempted by T_1. In such cases, we have:

$$v(\langle (n_1,n_2),A_1\rangle) = -V_3 r + X(n_1+1,n_2).$$

Theorem 1: If $f(n_1,n_2)$ is convex and increasing function on n_2 for any given n_1, the optimal policy is then a control limit policy. That means, for any state (n_1,n_2), there must exist an integer, say N_2, such that decision

$$a_{\langle (n_1,n_2),A_2\rangle} = \begin{cases} a_A, & \text{if } n_2 \le N_2, \\ a_R, & \text{if } n_2 > N_2. \end{cases} \tag{5}$$

Proof: If all VMs are busy when an SU arrives at state (n_1,n_2), we know that $C_1(n_1)+n_2 \ge C$ and then

$$C_2(n_1,n_2) = C - C_1(n_1).$$

Therefore

$$\begin{aligned} & \beta_0(n_1,n_2+2) = \beta_0(n_1,n_2+1) = \beta_0(n_1,n_2) \\ = \;& \lambda_1+\lambda_2+C_1(n_1)\mu_1+(C-C_1(n_1))\mu_2, \end{aligned} \tag{6}$$

For any two-dimensional integer function $g(n_1,n_2)$ ($n_1 \ge 0, n_2 \ge 0$), we introduce the following definitions for n_1 and n_2, respectively:

$$\Delta_{n_2} g(n_1,n_2) = g(n_1,n_2+1) - g(n_1,n_2). \tag{7}$$
$$\Delta^{(2)}_{n_2} g(n_1,n_2) = \Delta_{n_2} g(n_1,n_2+1) - \Delta_{n_2} g(n_1,n_2). \tag{8}$$

From the observation in equation (6), we will have

$$\begin{aligned} & \big(\alpha+\beta_0(n_1,n_2+1)\big)\Delta_{n_2} X(n_1,n_2) \\ = \;& -\Delta_{n_2} f(n_1,n_2) \\ & +\lambda_1\Delta_{n_2} v(\langle (n_1,n_2),A_1\rangle) + \lambda_2\Delta_{n_2} v(\langle (n_1,n_2),A_2\rangle) \\ & +C_1(n_1)\mu_1\Delta_{n_2} X(n_1-1,n_2) \\ & +(C-C_1)\mu_2\Delta_{n_2} X(n_1,n_2-1). \end{aligned} \tag{9}$$

By a similar implementation on equation (9) and by using the results in equations (6) , we have

$$\begin{aligned} & (\alpha+\beta_0(n_1,n_2+2))\Delta^{(2)}_{n_2} X(n_1,n_2) \\ = \;& -\Delta^{(2)}_{n_2} f(n_1,n_2) \\ & +\lambda_1\Delta^{(2)}_{n_2} v(\langle (n_1,n_2),A_1\rangle) + \lambda_2\Delta^{(2)}_{n_2} v(\langle (n_1,n_2),A_2\rangle) \\ & +C_1(n_1)\mu_1\Delta^{(2)}_{n_2} X(n_1-1,n_2) \\ & +(C-C_1)\mu_2\Delta^{(2)}_{n_2} X(n_1,n_2-1). \end{aligned} \tag{10}$$

We can now use Value Iteration Method with three steps to show that for all states $X(n_1,n_2)$ is concave and nonincreasing for nonnegative integer function on n_2 for any given n_1 as below:

Step 1: Set $X^{(0)}(n_1,n_2) = 0$, we know $v^{(0)}(\langle (n_1,n_2),A_2\rangle) = R$ and

$$v^{(0)}(\langle (n_1,n_2),A_1\rangle) = \begin{cases} -br, & n_1 < N_1, \\ 0, & n_1 = N_1. \end{cases}$$

Substitute these three results into equations, we will have

$$X^{(1)}(n_1,n_2) = \begin{cases} \frac{-f(n_1,n_2)+\lambda_1(-br)+\lambda_2 R}{\alpha+c}, & n_1 < N_1, \\ \frac{-f(n_1,n_2)+0+\lambda_2 R}{\alpha+c}, & n_1 = N_1. \end{cases}$$

Therefore, for any n_1, $X^{(1)}(n_1,n_2)$ is concave and nonincreasing on n_2.

Step 2: With these results in mind, and using the results in equations (9) and (10), we will know that

$$\Delta_{n_2} X^{(2)}(n_1,n_2) \le 0, \quad \text{and} \quad \Delta^{(2)}_{n_2} X^{(2)}(n_1,n_2) \le 0.$$

These two inequalities justify that for any n_1, $X^{(2)}(n_1,n_2)$ is nonincreasing and concave on n_2.

Step 3: Finally, by noting the *Theorem 11.3.2* of [15] that the optimality equation has the unique solution, we know the value iteration $X^{(n)}(n_1,n_2)$ will uniquely converges. Therefore, as the iteration continues, with n goes to ∞, with the property of nonincreasing and concave on n_2 for $X(n_1,n_2)$, it is straight forward to know that the optimal policy is a control limit policy as stated in the Theory.

The proof is now completed.

IV. Numerical Analysis

We have conducted a theoretical verification to demonstrate that the optimal policy for maximizing the total expected discount reward, expressed in equation (2), is a control limit policy or a threshold policy for accepting T_2 arrivals. The parameter values as listed in Table II, which are comparable to those reported in reference [18]. With this parameter setting,

TABLE II
Parameters Set

C	20
α	0.5
λ_i	$\lambda_1 = 1$, $\lambda_2 = 2$
μ_i	$\mu_1 = 6$, $\mu_2 = 8$
b	5
r	3
R	5
$f(n_1, n_2)$	$n_1^2 + 3n_2^2$

by using value iteration method we can derive both the $X(n_1, n_2)$ Values and the corresponding optimal policy, which are listed in the following tables.

TABLE III
Actions for T_2 task of Optimal Policy

	0			$n_2 \rightarrow$		5
0	1	1	1	1	1	1
	1	1	1	1	1	1
$n_1 \downarrow$	1	1	1	1	1	1
	1	1	1	1	1	1
4	1	1	1	1	1	1
	6			$n_2 \rightarrow$		11
0	1	1	1	1	1	0
	1	1	1	1	0	0
$n_1 \downarrow$	1	1	1	0	0	0
	1	1	0	0	0	0
4	1	0	0	0	0	0

According to Table III, the action code "1" indicates the acceptance, and the code "0" indicates that the system will refuse. Since the reward R is far beyond the holding cost and rejection cost, it can be seen from the table that the system will still accept the T_2 task to the buffer even if some T_2 task is waiting in the buffer, or if there is T_1 task is waiting in the buffer. As shown in Table IV, the value of function $X(n1, n2)$ decreases in the n_2 direction, which is consistent with our theoretical results.

V. Conclusion and Discussion

In order to make better use of the limited VM resources in private DCs, this paper proposes a new scheme: priority tasks have high priority and can interrupt non-priority tasks; non-priority tasks are scientific computing tasks and can occupy all the unused VM resources of the priority tasks. Serving for non-priority tasks is rewarded, and holding and disrupting these tasks is costly. To obtain the best total expected discount reward, the problem is formulated as a CTMDP model and verifies that the optimal policy for accepting non-priority tasks is a state-dependent control limit (threshold) policy.

TABLE IV
$X(n_1, n_2)$ Values with Optimal Policy

	0			$n_2 \rightarrow$		5
0	1.05	0.82	0.47	-0.08	-0.90	-2.07
	1.00	0.64	0.15	-0.59	-1.69	-3.22
$n_1 \downarrow$	0.93	0.51	-0.15	-1.16	-2.63	-4.65
	0.77	0.29	-0.54	-1.83	-3.67	-6.17
4	0.49	-0.06	-1.05	-2.61	-4.83	-7.81
	6			$n_2 \rightarrow$		11
0	-3.67	-5.79	-8.49	-11.85	-15.96	-20.88
	-5.29	-7.96	-11.32	-15.44	-20.39	-26.24
$n_1 \downarrow$	-7.29	-10.65	-14.80	-19.79	-25.71	-32.62
	-9.41	-13.45	-18.37	-24.23	-31.11	-39.09
4	-11.62	-16.33	-21.99	-28.70	-36.53	-45.56
	12			$n_2 \rightarrow$		17
0	-26.66	-33.40	-41.17	-50.04	-60.09	-71.39
	-33.06	-40.94	-49.95	-60.16	-71.65	-84.51
$n_1 \downarrow$	-40.62	-49.76	-60.14	-71.83	-84.90	-99.43
	-48.25	-58.67	-70.42	-83.58	-98.23	-114.44
4	-55.88	-67.55	-80.65	-95.26	-111.45	-129.31

References

[1] C. Kotas, T. Naughton, and N. Imam, "A comparison of amazon web services and microsoft azure cloud platforms for high performance computing," in *2018 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2018, pp. 1–4.

[2] P. Prukkantragorn and K. Tientanopajai, "Price efficiency in high performance computing on amazon elastic compute cloud provider in compute optimize packages," in *2016 International Computer Science and Engineering Conference (ICSEC)*, Dec 2016, pp. 1–6.

[3] H. Artail, M. A. R. Saghir, M. Sharafeddin, H. Hajj, A. Kaitoua, R. Morcel, and H. Akkary, "Speedy cloud: Cloud computing with support for hardware acceleration services," *IEEE Transactions on Cloud Computing*, vol. 7, no. 3, pp. 850–865, 2019.

[4] X. Hu, L. Wang, K. Wong, M. Tao, Y. Zhang, and Z. Zheng, "Edge and central cloud computing: A perfect pairing for high energy efficiency and low-latency," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 1070–1083, 2020.

[5] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp. 931–945, 2011.

[6] T. S. Toland, "C2cloud: A cloud computing framework," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan 2017, pp. 1–7.

[7] J. Mei, K. Li, and K. Li, "Customer-satisfaction-aware optimal multi-server configuration for profit maximization in cloud computing," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 1, pp. 17–29, 2017.

[8] S. Tayeb, M. Mirnabibaboli, L. Chato, and S. Latifi, "Minimizing energy consumption of smart grid data centers using cloud computing," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan 2017, pp. 1–5.

[9] Y. Shi, K. Suo, J. Hodge, D. P. Mohandoss, and S. Kemp, "Towards optimizing task scheduling process in cloud environment," in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan 2021, pp. 0081–0087.

[10] W. Ni, Y. Zhang, and W. Li, "Task admission control and boundary analysis of cognitive cloud data centers," 2020. [Online]. Available: https://arxiv.org/abs/2010.02457

[11] Q. Li, L. Zhao, J. Gao, H. Liang, L. Zhao, and X. Tang, "Smdp-based coordinated virtual machine allocations in cloud-fog computing systems," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1977–1988, June 2018.

[12] L. A. Barroso, J. Clidaras, and U. Hoelzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool, 2013.

[13] S. Sharif, P. Watson, J. Taheri, S. Nepal, and A. Y. Zomaya, "Privacy-aware scheduling saas in high performance computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1176–1188, 2017.

[14] W. Ni, Y. Zhang, and W. W. Li, “An optimal strategy for resource utilization in cloud data centers,” *IEEE Access*, vol. 7, pp. 158 095–158 112, Oct 2019.

[15] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Mar 2005.

[16] O. Alagoz and M. Ayvaci, *Uniformization in Markov Decision Processes*. Wiley Encyclopedia of Operations Research and Management Science, 2011.

[17] S. Ross, *Stochastic Processes*, Jan 1995.

[18] H. Khazaei, J. Misic, and V. B. Misic, “A fine-grained performance model of cloud computing centers,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 11, pp. 2138–2147, 2013.