Optimal Admission Policy In a Cloud Data Center With Priority and Non-Priority Tasks

Wenlong Ni, Yuhong Zhang, and Wei Li

Texas Southern University, 3100 Cleburne St, Houston, USA TX 77004,
wenlong.ni@tsu.edu, yuhong.zhang@tsu.edu, wei.li@tsu.edu

Abstract. This paper studies a cloud datacenter (DC) consisting of two types of tasks with different priority levels. While non-priority tasks generally request the use of a single virtual machine (VM), priority tasks may utilize multiple available VMs to accelerate processing. We focus on determining whether to accept or reject non-priority tasks to maximize overall system benefits. By formulating the problem as a stochastic dynamic program, t is verified that the best approach for handling non-priority tasks adheres to a control-limit framework. Both experimental outcomes and numerical evaluations highlight the efficacy of the proposed method, leading to the identification of the optimal threshold. The key contribution of this paper is the development of a stochastic dynamic program for DC resource management and the explicit derivation of an optimal control-limit policy. Both value iteration and linear programming methods are utilized to solve optimization problems. These results offer essential understanding for assessing the performance of various DC models, optimizing both rewards and resources efficiently.

Keywords: Cloud Data Center, Resource Utilization, Optimal Policy, Cost and Reward, Stochastic Dynamic Program

1 Introduction

Authors in papers [1–4] studied various cloud computing paradigms that delivers dynamically elastic and virtualized resources over the Internet. It functions as a core framework for data services, integrating networks, computational resources, storage systems, and software elements. The term "cloud" symbolizes the focus on resource utilization rather than the underlying implementation mechanisms. In CC, the cost for computation and energy consumption is less at current cloud DCs [5] because it exploits virtualization technology, which segregates the elementary functions of computers from the hardware resources and the physical infrastructure.

CC leverages virtualization technology [6–9] to create virtual machines (VMs) which uses hypervisors to abstract physical hardware, allowing multiple VMs

* This work has been funded in part by the National Science Foundation through awards numbered 2302469 and 2318662, alongside support from NASA under Grant 80NSSC22KM0052.

to running independently on a single server. This technology supports cloud environments by optimizing hardware usage, reducing operational costs, and enabling rapid deployment of services, making it a cornerstone of modern IT infrastructure.

The primary goal of a cloud service provider (CSP) is to maximize profitability while ensuring adherence to key performance indicators and service level agreements [10]. By efficiently managing infrastructure, leveraging automation, and implementing dynamic pricing models, operators aim to balance customer satisfaction with revenue growth. Additionally, they focus on reducing energy consumption and improving resource utilization to enhance sustainability and cost-effectiveness. Ultimately, the operator's success lies in achieving a competitive edge by delivering high-quality services at optimal costs, ensuring long-term profitability and customer retention.

Cloud computing (CC) has revolutionized large-scale big data processing and intricate computational analysis [11, 12]. Due to the underutilization of data center (DC) power [13] and the growing demand for scientific computations [11, 14], this study introduces an innovative resource allocation strategy for DCs to enhance the efficiency of virtual machine (VM) resource usage. The standard operations of a DC are categorized as priority tasks, while computationally intensive tasks are classified as non-priority. The system exhibits several unique features [10, 15], including task categorization, urgency levels, resource optimization, a preemption mechanism, and task queuing.

In one of our previous work [16], we investigated a general scenario without cognitive characteristics. We studied both an average-reward model and a discounted-expected-reward model. The key difference between the model in that paper and the one in the current study is the absence of task prioritization. Furthermore, in the model presented in [17], a task T_2 will utilize as many VMs as possible, whereas in the proposed model, each T_2 task occupies only one VM. The primary objective of this paper is to optimize the processing of non-priority tasks and identify an optimal policy that maximizes the total expected discounted reward for every initial state. Some more related ideas and notation discussed in the paper can be found in our unpublished work [18], which includes more theoretic details. Below, we outline the major contributions of this study:

1. Separate buffers are allocated for two types of tasks with different priority in this model. When a priority requested a number of VMs task arrives, it may preempt a non-priority task that is currently being processed by using one VM. We demonstrate that the optimal strategy for determining whether to accept or decline a non-priority task follows a state-dependent threshold policy, also known as a control-limit policy.
2. Both the value iteration (VI) and linear programming (LP) methods are utilized to solve Bellman optimization problems. Our approach ensures that resource allocation is maximized while adhering to predefined constraints. The use of LP not only streamlines the optimization process, but also guarantees precise and actionable insights.

The paper is structured as follows. Section II presents the system model of the DC. Section III discusses the optimal policy framework aimed at maximizing the reward, including the verification process that confirms it as a control limit policy. Section IV provides a numerical analysis supported by tables and diagrams to validate the theoretical findings. The paper concludes with final remarks in Section V.

2 Model Description and Analysis

A CC environment offers users and various application systems the ability to obtain computing power, storage capacity, or virtual machine services on demand from a dynamically virtualized resource pool. It is a continuously operating and changing system, and thus a continuous-time Markov decision process (CTMDP) framework is well-suited for modeling dynamic stochastic processes. We begin by presenting a system model for a DC, incorporating the necessary assumptions for all relevant parameters. Subsequently, we outline the construction of key components within the constructed CTMDP model. We will consider the following assumptions in the proposed model:

1. The system handles two types of tasks: priority tasks, referred to as type-1 (T_1) tasks, and non-priority tasks, referred to as type-2 (T_2) tasks. The number of VMs (denoted by C) will serve both tasks fluctuates dynamically in response to the workload within the system.
2. Tasks of type T_1 are time-critical and require a predetermined number (say b, a positive integer) of VMs for their execution, while tasks of type T_2, which involve additional payment, can be processed using a regular VM.
3. The arrival of tasks T_1 and T_2 follows Poisson processes [19] with arrival rates λ_1 and λ_2, respectively. The processing time for these tasks on a single VM follows a negative exponential distribution with rates μ_1 and μ_2, respectively.

Based on these assumptions, we are now ready to construct a CTMDP model as follows:

1. Our emphasis is on decision-making states, which include both standard system states and events occurring at decision points. The conventional state, the first component of a decision-making state, is defined by the number of ongoing tasks of each type in the DC. This is denoted as $S = \{s : s = (n_1, n_2),\ n_1 \geq 0,\ n_2 \geq 0\}$, where n_1 and n_2 are the number of T_1 and T_2 tasks. The event space is defined by $e \in E = \{D_i, A_i, i = 1, 2\}$, where D_i indicates the departure of a T_i task from the system after service completion, and A_i signifies the arrival of a T_i task. Thus, a decision-making state can be expressed as $\hat{s} = \langle s, e\rangle = \langle (n_1, n_2), e\rangle$. The state space is the collection of all possible decision-making states, represented as $\hat{S} = S \times E = \{\hat{s} | \hat{s} = \langle (n_1, n_2), e\rangle\}$.
2. Once a service in progress completes, the controller remains inactive and makes no decision. We introduce the notation a_D to represent a hypothetical action corresponding to the completion (departure) of a service. Let a_A

denote the action to admit and a_R denote the action to reject the request. The action space A is defined as the set of three actions: $A = \{a_D, a_A, a_R\}$.

3. Per our assumption, the time duration between two epochs is exponentially distributed. Let $V_1(n_1)$ be the number of VMs occupied by T_1 tasks, $V_2(n_1, n_2)$ be the number of VMs occupied by T_2 tasks.

$$\begin{aligned} V_1(n_1) &= bn_1, n_1 < N_1, \\ V_1(n_1) &= C, n_1 \geq N_1, \\ V_2(n_1, n_2) &= \min(C - V_1(n_1), n_2), n_1 < N_1, \\ V_2(n_1, n_2) &= 0, n_1 \geq N_1. \end{aligned}$$

Denote by $s = (n_1, n_2)$ and $\beta_0(s) = \lambda_1 + \lambda_2 + V_1\mu_1 + V_2\mu_2$, we know that the average duration for the system to transition from state s to any other state is, is $1/\beta_0(s)$.

4. Let $q(j|\hat{s}, a)$ denote the probability that the system occupies state j in the next epoch if taking action a from state $\hat{s}$. For a event D_1 under the condition of $(n_1 > 0)$, $(\hat{s}, a) = (\langle (n_1, n_2), D_1 \rangle, a_D)$, if denote by $s_{d_1} = (n_1 - 1, n_2)$, then $q(j|\langle (n_1, n_2), D_1 \rangle, a_D)$ can be derived as

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

The transition probabilities for other states can also be derived similarly.

5. The reward function is involved with the income award $k(\hat{s}, a)$ and the system cost at rate $c(\hat{s}, a)$, and can be derived by:

$$r(\hat{s}, a) = k(\hat{s}, a) + \frac{c(\hat{s}, a)}{\alpha + \beta(\hat{s}, a)},$$

where

$$k(\hat{s}, a) = \begin{cases} R, e = A_2, a = a_A, \\ 0, \quad otherwise. \end{cases}$$

After accepting a T_2 task, the reward is received after the service completion, which is equal to putting it with the accept action. Let $f(s), s = (n_1, n_2)$ be the cost rate of state s, then $c(\hat{s}, a)$ fulfills the following conditions:

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

A policy specifies the decision rule to be used at every decision epoch. Our objective is to determine the optimal policy π that maximizes $v_\alpha^\pi(\hat{s})$ for all initial states $\hat{s}$.

3 Optimal Stationary State-Related Control Limit Policy

Furthermore, a policy is called a control limit policy (or a threshold policy) if there exists a threshold in the policy for accepting task arrivals. In this research, since we only focus on admitting T_2 task, when there are n_1 tasks of T_1 in the system, there is a threshold $T(n_1) \geq 0$ such that the system will only accept the **arriving** T_2 whenever the number of T_2 tasks currently in the system is less than $T(n_1)$, and reject the T_2 arrivals otherwise. This means the decision rule for T_2 tasks is:

$$d(n_1, n_2, A_2) = \begin{cases} a_A, n_2 \leq T(n_1), \\ a_R, n_2 > T(n_1). \end{cases} \tag{1}$$

It is easy to see that a threshold policy makes the choices for decision makers (CSP) very simple.

3.1 Optimal State Value Function

Let $V^*(s)$ denote the optimal state value function, which represents the maximum expected cumulative reward starting from state s and following the optimal policy π^* thereafter. The Bellman optimality equation for the state value function is given by:

$$V^*(s) = \max_{a \in A(s)} \left(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^*(s') \right),$$

where:

- $R(s,a)$ is the immediate reward for taking action a in state s.
- $\gamma \in [0,1)$ is the discount factor, which weighs future rewards.
- $P(s'|s,a)$ is the transition probability from state s to state s' given action a.

The optimal policy π^* can be derived by choosing the action that maximizes the right-hand side of the equation for each state s:

$$\pi^*(s) = \arg\max_{a \in A(s)} \left(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^*(s') \right).$$

3.2 Optimal Result By Value Iteration

By using above equation, for a departure event of D_1, we have

$$\begin{aligned} v(\langle (n_1+1, n_2), D_1 \rangle) = \frac{1}{\alpha + \beta_0(n_1,n_2)} [& -f(n_1,n_2) + \lambda_1 v(\langle (n_1,n_2), A_1 \rangle) \\ & + \lambda_2 v(\langle (n_1,n_2), A_2 \rangle) + n_1 \mu_1 v(\langle (n_1,n_2), D_1 \rangle) \\ & + n_2 \mu_2 v(\langle (n_1,n_2), D_2 \rangle)]. \end{aligned} \tag{2}$$

From above equations, it is seen that the values of $v(\hat{s})$ is mainly dependent on the number of n_1 and n_2, so we can define a new function $B(s), s = (n_1, n_2)$, which is

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

From these analysis, since there is only accept/reject actions for the T_2 arrivals, it is not too hard to verify that

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

For the T_1 tasks, we have

$$v(\langle (n_1, n_2), A_1 \rangle) = B(n_1 + 1, n_2).$$

For any two-dimensional integer function $X(n_1, n_2), n_1 \geq 0, n_2 \geq 0$, we introduce the following definitions:

$$\begin{aligned} \Delta_{n_2} X(n_1, n_2) &= X(n_1, n_2 + 1) - X(n_1, n_2). \\ \Delta^{(2)}_{n_2} X(n_1, n_2) &= \Delta_{n_2} X(n_1, n_2 + 1) - \Delta_{n_2} X(n_1, n_2). \end{aligned}$$

Theorem 1: If the cost function $f(n_1, n_2)$ is convex and increasing function on n_2 for any given n_1, which means $\Delta_{n_2} f(n_1, n_2) \geq 0$ and $\Delta^{(2)}_{n_2} f(n_1, n_2) \geq 0$, the optimal policy for admitting T_2 arrivals is then a control limit policy.

Proof: For any $n_2 > 0$, we can easily get

$$\beta_0(n_1, n_2) = \begin{cases} \lambda_1 + \lambda_2 + b n_1 \mu_1 + V_2(n_1, n_2)\mu_2, & n_1 < N_1, \\ \lambda_1 + \lambda_2 + C\mu_1, & n_1 \geq N_1, \end{cases} \tag{3}$$

Furthermore, by using the notation of $B(n_1, n_2)$, we can rewrite the equation (2) as below:

1. If $n_1 < N_1$,

$$\begin{aligned} B(n_1, n_2) = \frac{1}{\alpha + \beta_0(n_1, n_2)} \Big[& - f(n_1, n_2) \\ & + \lambda_1 v(\langle (n_1, n_2), A_1 \rangle) + \lambda_2 v(\langle (n_1, n_2), A_2 \rangle) \\ & + b n_1 \mu_1 B((n_1 - 1, n_2)) \ + \ V_2(n_1, n_2)\mu_2 B((n_1, n_2 - 1)) \Big]. \end{aligned} \tag{4}$$

2. If $n_1 \geq N_1$,

$$\begin{aligned} B(n_1, n_2) = \frac{1}{\alpha + \beta_0(n_1, n_2)} \Big[& - f(n_1, n_2) + \lambda_1 v(\langle (n_1, n_2), A_1 \rangle) \\ & + \lambda_2 v(\langle (n_1, n_2), A_2 \rangle) + C\mu_1 B((n_1 - 1, n_2)) \Big]. \end{aligned} \tag{5}$$

From the observation in equation (3), (4) and the equation (5), we will have

1. If $n_1 < N_1$,

$$\begin{aligned}&\big(\alpha+\beta_0(n_1,n_2+1)\big)\Delta_{n_2}B(n_1+1,n_2)\\ =&-\Delta_{n_2}f(n_1,n_2)+\lambda_1\Delta_{n_2}v(\langle(n_1,n_2),A_1\rangle)+\lambda_2\Delta_{n_2}v(\langle(n_1,n_2),A_2\rangle)\\ &+bn_1\mu_1\Delta_{n_2}B(n_1,n_2)+V_2(n_1,n_2)\mu_2\Delta_{n_2}B(n_1,n_2-1).\end{aligned}\tag{6}$$

2. If $n_1 \geq N_1$,

$$\begin{aligned}&\big(\alpha+\beta_0(n_1,n_2+1)\big)\Delta_{n_2}B(n_1,n_2)\\ =&-\Delta_{n_2}f(n_1,n_2)+\lambda_1\Delta_{n_2}v(\langle(n_1,n_2),A_1\rangle)\\ &+\lambda_2\Delta_{n_2}v(\langle(n_1,n_2),A_2\rangle)+C\mu_1\Delta_{n_2}B(n_1,n_2).\end{aligned}\tag{7}$$

By a similar implementation with above equations (6) and (7), we have

1. If $n_1 < N_1$,

$$\begin{aligned}&(\alpha+\beta_0(n_1,n_2+2))\Delta^{(2)}_{n_2}B(n_1,n_2)\\ =&-\Delta^{(2)}_{n_2}f(n_1,n_2)\\ &+\lambda_1\Delta^{(2)}_{n_2}v(\langle(n_1,n_2),A_1\rangle)+\lambda_2\Delta^{(2)}_{n_2}v(\langle(n_1,n_2),A_2\rangle)\\ &+bn_1\mu_1\Delta^{(2)}_{n_2}B(n_1-1,n_2)+V_2(n_1,n_2)\mu_2\Delta^{(2)}_{n_2}B(n_1,n_2-1).\end{aligned}\tag{8}$$

2. If $n_1 \geq N_1$,

$$\begin{aligned}&(\alpha+\beta_0(n_1,n_2+2))\Delta^{(2)}_{n_2}B(n_1,n_2)\\ =&-\Delta^{(2)}_{n_2}f(n_1,n_2)+\lambda_1\Delta^{(2)}_{n_2}v(\langle(n_1,n_2),A_1\rangle)\\ &+\lambda_2\Delta^{(2)}_{n_2}v(\langle(n_1,n_2),A_2\rangle)+C\mu_1\Delta^{(2)}_{n_2}B(n_1-1,n_2).\end{aligned}\tag{9}$$

With the preparations on all equations from equation (8) to (9), we can now adopt Value Iteration Method with three steps to prove that for any given n_1, the values of $B(n_1,n_2)$ is concave and nonincreasing on n_2 as below:

Define $B^{(0)}(n_1,n_2)=0$ as the value of $B((n_1,n_2)$ in the initial $(0th)$ iteration and $v^{(0)}$ being the corresponding v, we know $v^{(0)}(\langle(n_1,n_2),A_2\rangle)=R$ and $v^{(0)}(\langle(n_1,n_2),A_1\rangle)=0$. Next, define $B^{(1)}(n_1,n_2)$ as the value of $B((n_1,n_2)$ in the $(1st)$ iteration, we will have

$$B^{(1)}(n_1,n_2)=\frac{-f(n_1,n_2)+\lambda_2 R}{\alpha+c}$$

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

By using above concavity and non-increasing property of $B^{(1)}(n_1,n_2)$, let $v^{(1)}$ be the corresponding v in the $(1st)$ iteration, we know that $v^{(1)}(\langle n_1,n_2,A_1\rangle)$ is concave and non-increasing functions for any n_2. By further applying the result in Lemma 1 of [16], we know that $v^{(1)}(\langle n_1,n_2,A_2\rangle)$ is also concave and non-increasing functions for any n_2. With these results in mind, and using the results in equations (6), (7) and (8), (9) we will know that

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

These two inequalities justify that for any n_1, $B^{(2)}(n_1, n_2)$ is nonincreasing and concave on n_2. Here, $B^{(2)}(n_1, n_2)$ is the value of $B((n_1, n_2)$ in the $(2nd)$ iteration.

Finally, by noticing the *Theorem 11.3.2* of [20] that the optimality equation has the unique solution, we know the value iteration $B^{(i)}(n_1, n_2), (i = 0, 1, \ldots,)$ will uniquely converges. Therefore, as the iteration continues, with n goes to ∞, for any n_1, $B(n_1, n_2)$ is always concave nonincreasing for any n_2.

Remark: Through the verification process for Theorem 1, it is observed that the threshold for accepting T_2 tasks exists regardless the number of VMs C. This observation fits the fact that the available VMs in a DC may be constantly changing due to dynamic loads. Generally speaking, if the number of VMs C is larger, the processing speeds are higher, so the DC can accept more T_2 tasks waiting in the buffer.

3.3 Optimal Result By Linear Programming

In addition to the Value Iteration method, the Bellman optimality equations can also be formulated as a linear programming problem to find the optimal value function $V^*(s)$. Using the linear programming method, it systematically tackles the optimization problem, aiming to identify the most efficient strategy for resource allocation or decision making. The integration of the threshold policy with linear programming provides a robust analytical approach, enhancing the understanding and practical application of the model in various contexts.

A linear programming problem consists of three main components:

- **Objective Function**: A linear function to be maximized or minimized.
- **Constraints**: A set of linear inequalities or equalities that define the feasible region.
- **Decision Variables**: Variables that represent the choices available to the decision maker.

In matrix notation, the LP problem can be written as follows:

Objective Function:

Maximize or Minimize $\quad Z = c^T\mathbf{x}$ where $\mathbf{c}$ is the vector of coefficients, and $\mathbf{x}$ is the vector of decision variables.

Subject to Constraints:

$$\mathbf{Ax} \leq \mathbf{b}$$

where $\mathbf{A}$ is the matrix of constraint coefficients, and $\mathbf{b}$ is the vector of bounds.

Non-negativity:

$$\mathbf{x} \geq 0$$

For the Bellman equation of the MDP model proposed in this paper, the linear programming formulation is given by:

$$Minimize \quad \sum_{s \in S} V(s),$$

subject to the constraints:

$$V(s) \geq R(s,a) + \sum_{s' \in S} P(s'|s,a)V(s') \quad \forall s \in S,\ a \in A(s).$$

In this formulation:

- $V(s)$ denotes the approximate value assigned to state s.
- The objective function $\sum_{s \in S} V(s)$ is to minimize the aggregate of state values, thereby aligning closely with the optimal value function $V^*(s)$ within the given constraints.

This linear programming technique offers an alternative strategy for tackling Bellman optimality equations, particularly advantageous for large or intricate MDPs where finding exact solutions is computationally demanding.

4 Numerical Analysis

In this section, we show the threshold policy numerically considering specific parameters as indicated in Table 1:

Table 1. Parameters Selection

Number of VMs C	20
Discount Factor α	0.1
λ_1 / μ_1	0.5 / 3
λ_2 / μ_2	1 / 4
b	2
Reward R	3
Cost Function $f(n_1, n_2)$	$n_1^2 + n_2^2$

4.1 Value Iteration Method

Using this parameter configuration, we can determine both the $B(n_1, n_2)$ values and the corresponding optimal policy using the value iteration method. The results are presented in the tables below.

Table 2 illustrates that the values of the function $B(n_1, n_2)$ exhibit a concave decreasing pattern as n_2 increases, which aligns with our theoretical expectations. To show how the threshold values depend on the value of n_1, by choosing $b = 2$, we plot two different cases when $n_1 = 4$, and $n_1 = 8$, respectively, in the Fig. 1.

In this Figure, the green color is for the value of $R + B(n_1, n_2 + 1)$ and the red color is for the value of $B(n_1, n_2)$. Since $B(n_1, n_2)$ is concave on n_2 for any given n_1, and thus $R + B(n_1, n_2 + 1)$, it is easy to identify the optimal threshold

Table 2. $B(n_1, n_2)$ Values with Optimal Policy

	$n_2 = 0$	1	2	3	4	5	6	7	8	9	10	11
$n_1 = 0$	26.07	25.77	25.22	24.42	23.37	22.08	20.54	18.76	16.72	14.45	11.92	9.15
1	25.89	25.59	25.04	24.24	23.20	21.90	20.37	18.58	16.55	14.27	11.74	8.97
2	25.55	25.25	24.70	23.90	22.85	21.56	20.02	18.24	16.20	13.92	11.40	8.63
3	25.04	24.74	24.19	23.39	22.34	21.05	19.51	17.73	15.70	13.42	10.89	8.12
4	24.37	24.07	23.51	22.72	21.67	20.38	18.84	17.06	15.02	12.74	10.22	7.44
5	23.53	23.23	22.68	21.88	20.83	19.54	18.00	16.22	14.18	11.90	9.37	6.44
6	22.53	22.22	21.67	20.87	19.83	18.54	17.00	15.21	13.18	10.76	8.00	4.84
7	21.36	21.05	20.50	19.70	18.66	17.37	15.83	13.93	11.71	9.10	6.12	2.71
8	20.02	19.72	19.17	18.37	17.32	15.95	14.27	12.22	9.81	6.98	3.74	0.05

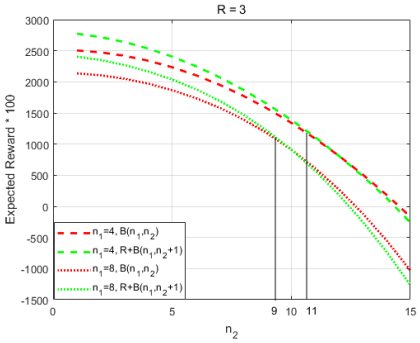

Fig. 1. Optimal Threshold Values.

by comparing the red line ($B(n_1, n_2)$) and the green line ($R + B(n_1, n_2 + 1)$), as shown in Fig. 1 if the green line is over the red line which means the system would take the accept action, so the threshold is 11 for $n_1 = 4$, to be 9 for $n_1 = 8$, respectively. Next, it is also a straightforward observation that the optimal threshold is therefore a decreasing function of n_1.

4.2 Linear Programming

Similar to the Value Iteration method, to confirm the effectiveness of the proposed Linear Programming (LP) model, we performed a series of numerical experiments to assess its performance with various parameters of the system. This linear programming approach provides an alternative method for solving Bellman optimality equations, especially useful for large or complex MDPs where exact solutions are challenging to compute.

As illustrated in Table 3, the numeral "1" denotes acceptance of a task, whereas "0" indicates rejection by the system. Given the significant disparity between the reward R and the associated holding and rejection costs, the table reveals that the system tends to accept T_2 tasks into the buffer, regardless of whether there are already waiting T_2 tasks or even T_1 tasks present. It can be

Table 3. Actions for T_2 task of Linear Programming Solution

	$n_2=0$	1	2	3	4	5	6	7	8	9	10	11
$n_1=0$	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	0
7	1	1	1	1	1	1	1	1	1	1	1	0
8	1	1	1	1	1	1	1	1	1	1	0	0

easily found that the actions in Table 3 are the same as those derived from the values in Table 2.

These experiments and data analysis have clearly demonstrated the effectiveness of the proposed method. Through rigorous analysis, we have identified the optimal threshold, which significantly enhances performance. Furthermore, we have observed a discernible pattern in how this threshold varies with different parameters. This discovery not only validates our methods but also provides valuable insights into the dynamic behavior of the system. Our findings pave the way for further refinement and optimization of the method, ensuring its robustness and applicability in various contexts.

5 Conclusion and Discussion

To optimize VM usage in resource-limited DCs, this paper proposes a scheme for handling both priority and non-priority tasks. Priority tasks preempt non-priority scientific computing tasks, which utilize available VM resources. Serving non-priority tasks generates rewards, while holding or interrupting them incurs costs. We formulated this as a CTMDP model and identified the optimal policy for admitting non-priority tasks to be a state-dependent threshold policy. Furthermore, the use of LP enables the efficient formulation and solution of complex optimization problems. LP provides a powerful framework for solving task scheduling problems in CC, and the results of this study suggest that LP can be a valuable tool for improving efficiency and effectiveness. By continuing to explore and refine LP models, we can develop more robust optimization strategies that can be applied to real-world cloud data center, ultimately leading to better outcomes for both users and service providers. Through the integration of reward for task acceptance and holding cost for task processing, the MDP model provides a flexible framework for optimizing task scheduling in cloud environments. Using a Markov Decision Process with linear programming for optimization, this model balances load distribution, minimizes delay costs, and ensures efficient VM utilization. By balancing these factors, we achieve a net increase in system efficiency, which enables a more dynamic approach to resource

allocation under varying load conditions. The findings of this paper can serve as an economically optimal strategy in diverse cloud data centers (DCs). Our future research aims to derive optimal system policies for maximizing rewards, even with incomplete or partial system information, by incorporating advanced machine learning techniques and other methodologies.

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)*, pages 1–4, Jan 2018.
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)*, pages 1–6, Dec 2016.
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*, 7(3):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*, 19(2):1070–1083, 2020.
5. Z. Cao, X. Zhou, X. Wu, Z. Zhu, T. Liu, J. Neng, and Y. Wen. Data center sustainability: Revisits and outlooks. *IEEE Transactions on Sustainable Computing*, 9(3):236–248, 2024.
6. H. Jin, D. Pan, J. Xu, and N. Pissinou. Efficient vm placement with multiple deterministic and stochastic resources in data centers. In *GLOBECOM - IEEE Global Telecommunications Conference*, pages 2505–2510, Dec 2012.
7. Z. Xiao, W. Song, and Q. Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1107–1117, 2013.
8. Y. Yamato, Y. Nishizawa, S. Nagao, and K. Sato. Fast and reliable restoration method of virtual resources on openstack. *IEEE Transactions on Cloud Computing*, 6(2):572–583, 2018.
9. N. Kamiyama. Trading virtual machines to stabilize revenue in public clouds. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–8, Apr 2018.
10. J. Mei, K. Li, and K. Li. Customer-satisfaction-aware optimal multiserver configuration for profit maximization in cloud computing. *IEEE Transactions on Sustainable Computing*, 2(1):17–29, 2017.
11. 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*, 22(6):931–945, 2011.
12. B. Liang and J. Bai. Low-energy resource classification algorithm for cross-regional cloud data centers based on k-means clustering algorithm. *IEEE Transactions on Industrial Informatics*, 20(8):10084–10091, 2024.
13. 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.

14. Y. Chen, Z. Zhang, Y. Deng, G. Min, and L. Cui. A combined trend virtual machine consolidation strategy for cloud data centers. *IEEE Transactions on Computers*, 73(9):2150–2164, 2024.
15. 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*, 28(4):1176–1188, 2017.
16. W. Ni, Y. Zhang, and W. W. Li. An optimal strategy for resource utilization in cloud data centers. *IEEE Access*, 7:158095–158112, Oct 2019.
17. W. Ni, Y. Zhang, and W. W. Li. Optimal task admission control of private cloud data centers with limited resources. In *2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0167–0172, 2024.
18. W. Ni, Y. Zhang, and W. W. Li. Task admission control and boundary analysis of cognitive cloud data centers, 2020, https://arxiv.org/abs/2010.02457.
19. S. Ross. *Stochastic Processes*. Jan 1995.
20. M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Mar 2005.