Energy Consumption Optimization in Data Center with Latency Based on Histograms and Discrete-Time MDP

Léa Bayati
Laboratoire d'Algorithmique Complexité et Logique, Université Paris-Est Créteil (UPEC),

Keywords: Data Center, Energy Saving, Markov Decision Process, Quality of Services, Latency.

Abstract: This article introduces a probabilistic model for managing power dynamically (DPM) in a data center. The model involves switching servers on and off, while considering both the time it takes for the machines to become active and the amount of energy they consume. The goal of DPM is to balance energy consumption with Quality of Service (QoS) requirements. To construct the model, job arrivals and service rates are represented using histograms, which are discrete distributions derived from actual traces, empirical data, or measurements of incoming traffic. The data center is modeled as a queue, and the optimization problem is formulated as a discrete-time Markov decision process (MDP) in order to identify the optimal policy. The proposed approach is evaluated using real traffic traces from Google, and different levels of latency are compared.

1 INTRODUCTION

The recent expansion of *Clouds* and *Data Centers* are arising energetic and digital pollution challenges. Energy consumed by data centers is estimated to be more than 1.3% of the global energy consumption. Otherwise, the daily CO_2 emission of one data center server can be evaluated to more than 10 kg. Those rates are increasing and needs for considering power management strategy is emerging face of the energetic problems and digital pollution issues (Rajesh et al., 2008). Data centers are conceived to hold up the peak traffic load, however the global load is less than three quarters of the peak load (Benson et al., 2010). Thus, a considerable number of machines are not under load and still consume more than the half of the maximal energy consumption (Greenberg et al., 2009). Papers like (Lee and Zomaya, 2012) show that 70% of the total cost of the data center is spent for electricity used to run the servers and to make them cooler. Therefore, the primary determinant of energy consumption is the number of active servers. To reduce energy usage, a power management strategy can be implemented to control the on/off status of servers in a data center, which ensures both efficient performance of the services provided by the data center and reasonable energy consumption. Two requirements are in conflict: (i) Saving energy, and (ii) Increasing the Quality of the Service (QoS). To conserve energy, it is necessary to activate a limited number of servers, resulting in lower energy consumption but longer waiting times and a higher rate of job loss. Conversely, in order to improve QoS, more servers must be activated, which consumes more energy, but results in shorter waiting times, a lower rate of job loss, and higher energy consumption. Additionally, turning-on a server is not instantaneous and may cause an extra amount of energy. The objective, therefore, is to develop power management algorithms that consider both of these constraints to minimize waiting times, job loss rates, and energy consumption. In other word, it is necessary to define a cost for each requirement and conceive a policy that minimizes the overall cost. Notice that each job arriving to a data center may generate a profit. Works like in (Dyachuk and Mazzucco, 2010) suggest that a lost job may cost 6.2×10^{-6}\$. Otherwise, according to research published by Dell, Principled Technologies and other works as in (Rajesh et al., 2008), running, one server costs around 300$ per year.

2 RELATED WORK

Many works as in (Gandhi et al., 2013; Aidarov et al., 2013; Mitrani, 2013) model data centers as a theoretical queuing model to evaluate the trade-off between waiting time and energy consumption where jobs arrive according to Poisson process then served accord-

Bayati, L.
Energy Consumption Optimization in Data Center with Latency Based on Histograms and Discrete-Time MDP.
DOI: 10.5220/0011846400003491
In *Proceedings of the 12th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS 2023)*, pages 98-105
ISBN: 978-989-758-651-4; ISSN: 2184-4968

ing to an exponentially distributed service time. They consider specific management mechanisms to minimize energy consumption that leads to suboptimal strategies like threshold policy. By conducting experimental simulations and using analytical equations, they were able to show that it is possible to achieve a significant reduction in energy consumption (ranging from 20% to 40%) without sacrificing a reasonable waiting time. However works as (Maccio and Down, 2015; Bayati, 2018) were interested in finding the optimal policy by formalizing the energy saving models by *Markov Decision Process*.

MDP (Markov Decision Process) is a model that allows for the formalization of stochastic decision systems. It is based on a Markov chain that is augmented by a set of decision-making actions. Each action changes the system from one state to another and incurs a certain cost. Thus, the probability of transitioning to a new state is influenced by the selected action. In the context of power management, the cost of consumed energy depends on the number of active servers and the switching-on rate of servers. On the other hand, the cost of QoS depends on the number of jobs that are waiting and/or rejected. Furthermore, to find the optimal policy that minimizes the expected total cost accumulated over a finite period of time, all possible ways of turning off/turning on each machine from every state of the system must be considered.

Notice that most of studies consider the latency of servers, time latency (period needed to switch on/off a server) as in (Mitrani, 2013; Xu and Tian, 2008), or energetic latency (additional energy needed to switch on/off a server) as in (Schwartz et al., 2012; Dyachuk and Mazzucco, 2010; Leng et al., 2016), or both, as in (Maccio and Down, 2015; Gandhi et al., 2013; Entezari-Maleki et al., 2017), in the case of continuous-time framework, where arrival jobs are modeled by a continuous time distribution obtained by fitting the empirical data or by asserting some assumptions that leads usually to a Poisson distribution (Maccio and Down, 2015). Less work was addressed to the context of discrete-time MDP, were arrival jobs and service rates can be modeled by more general distributions (Benini et al., 1999).

More recent works can be found in (Chen and Wang, 2022; Khan, 2022; Rteil et al., 2022; Peng et al., 2022; An and Ma, 2022).

In fact, in the following we will show how MDP can be used to include the latency of servers. Notice that modeling our energetic optimization problem with an MDP when including latency is not easy. Thus, we will assume following assumptions: (i) only latency of switching-on is considered; (ii) turning-off a running server is instantaneous, which means that switching-off latency is zero; (iii) all servers are homogeneous and have the same constant latency period, which is of a duration of k units of time; (iv) the data center is modeled by a discrete time queue, including the set of homogeneous servers with the same level of energy consumption and the same service capacity; (v) real incoming traffic traces are sampled then used directly to build empirical discrete distributions (histograms) to model job arrivals and service rates. After that, our problem of energy-QoS optimization is formulated by a discrete-time MDP then the value iteration algorithm that implements the Bellman's backup equations (Bellman, 1957) is used to compute the optimal policy.

The rest of this paper is organized as follows. Section 3 models the system by simple queue. Then, Section 6 formulates the optimization problem as discrete-time MDP. After that, in Section 8 we give experimental results when analyzing a system with arrivals modeled by discrete distribution obtained from real Google traffic traces (Wilkes, 2011).

3 MODEL DESCRIPTION

In this work we consider a queue model that operates in discrete time. This model is based on a batch arrival queue with a finite buffer capacity of size b. Arrival jobs and service rates are modeled using histograms, which are discrete probability distributions derived from actual traces. The number of jobs arriving at the data center during a time slot is represented by a histogram denoted $\mathcal{H}_A$, where $P_A(i)$ gives the probability of having i arrival jobs per slot. The data center comprises a maximum of max homogeneous servers, and during a single time unit, each server can process a certain number of jobs, which is modeled using a histogram denoted $\mathcal{H}_D$, where $P_D(d)$ gives the probability of processing d jobs. Each server can be in one of several states:

- stopped: switched-off
- in latency since 0 unit of time and needs k unit to be ready
- in latency since 1 unit of time and needs $k-1$ unit to be ready
- ...
- in latency since i unit of time and needs $k-i$ unit to be ready
- ...
- in latency since $k-1$ unit of time and needs 1 unit to be ready
- ready: switched-on

Each slot we need to know the number of servers grouped by states given above (see Figure 1). We consider the set of the following levels of latency:

$$\mathcal{L} = \{lat_\infty = stop, lat_k, lat_{k-1}, \ldots, lat_i, \ldots, lat_2, lat_1, ready = lat_0\} \quad (1)$$

where lat_i denotes that the server is within the switching-on period and need i slot to be completely ready to work. The number of servers of level $x \in \mathcal{L}$ is denoted by m_x.

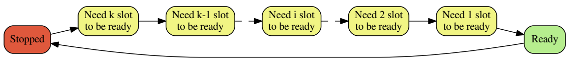

Figure 1: Transition between stopping, sleeping, and ready modes.

Thus, the number of operational servers is $m = m_{ready} = m_{lat_0}$, the number of servers that will be ready within i slot is m_{lat_i}, the total number of servers in latency is given by $m_{lat} = \sum_{i=1}^{k} m_{lat_i}$, and finally $max = m_{ready} + m_{lat} + m_{stop}$ is the maximal number of operational servers (total number of servers).

The number of jobs that are currently waiting in the queue is represented by the symbol n, while the number of jobs that have been rejected or lost is represented by l. At the start of the simulation, we assume that there are no waiting or rejected jobs, and all servers are in the "stopped" mode.

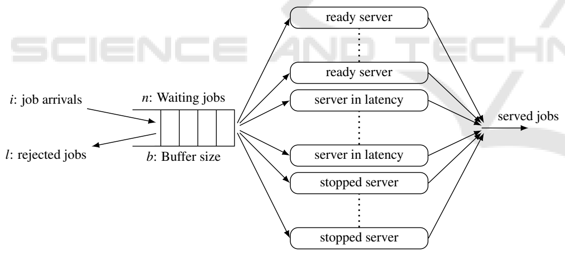

Figure 2: Illustration of the queuing model.

To compute the number of waiting jobs n, one can use an induction method where the specific sequence of events during a slot must be described. Initially, jobs are added to the buffer, then they are executed by the servers. Job admission is performed on a per-job basis using the Tail Drop policy, which means that a job is accepted if there is space in the buffer, otherwise it is rejected. The equations below show the number of waiting jobs in the buffer and the lost jobs, where i represents the number of arrival jobs, and d represents the service rate:

$$\begin{cases} n & \leftarrow & \min\{b, \max\{0, n+i-m_{ready} \times d\}\} \\ l & \leftarrow & \max\{0, n+i-m_{ready} \times d - b\} \end{cases} \quad (2)$$

Under the assumption that input arrivals are independent and identically distributed (i.i.d.), the queue model is represented by a time-homogeneous Discrete Time Markov Chain. However, when dealing with real traffic traces, it is necessary to ensure the i.i.d. assumption by finding the appropriate sampled period for which the sampled trace is i.i.d. So we use a statistical hypothesis testing called *the turning point test* (Kendall, 1973) to test the i.i.d-ness of the sampled trace, then consider only the sampled period for which the data is i.i.d.

Nonetheless, the system becomes increasingly complex to analyze as the number of servers varies over time. The number of servers may fluctuate based on traffic and performance metrics. To be more precise, n, l, m_{ready} are considered, then some decisions are taken depending on a particular cost function.

4 ENERGY AND PERFORMANCE METRIC

The energy consumption of the data center depends on the number of active servers. When a server is turned off, it doesn't consume any energy, but when it's operational, it consumes a certain amount of energy per time slot, which incurs a cost in monetary units (c_M). Additionally, since it takes k units of time for a server to switch on, it incurs an extra energy cost that also has a monetary value (c_{on}). We assume that the energy consumed during the latency period is uniform, i.e., $\frac{c_{on}}{k}$ per time slot. The total energy consumption is the sum of all the energy units consumed over a certain period. QoS depends on the number of jobs that are waiting or lost. Each waiting job incurs a cost in monetary units (c_N), while each rejected job has a different cost (c_L).

Table 1: Unitary costs of energy end Qos.

Cost	**Meaning**
$c_M \in \mathbb{R}^+$	energy cost for running one operational server
$c_{on} \in \mathbb{R}^+$	energy cost needed to switch-on a server
$c_N \in \mathbb{R}^+$	waiting cost for one job per unit of time
$c_L \in \mathbb{R}^+$	rejection cost of one lost job

5 OBJECTIVE COST FUNCTION

During a given period of time, a dynamic control policy for energy efficiency involves taking action in each slot (either turning on or turning off a specific number of servers) to adjust the number of operational servers in response to changes in incoming jobs. The optimal strategy is to find the best sequence of actions that minimizes the overall monetary cost, which is a combination of the energetic cost and the performance cost. The cost incurred for each slot can be expressed as:

$$n \times c_N + l \times c_L + m_{ready} \times c_M + m_{lat} \times \frac{c_{on}}{k} \quad (3)$$

where n is the number of waiting jobs, l is the number of lost jobs, m_{ready} is the current number of operational servers, and m_{lat} is the number of servers in latency mode.

6 MARKOV DECISION PROCESS

To analyze the performance and energy consumption of the data center under the optimal strategy, we will utilize the concept of a *Markov Decision Process* to formulate our optimization problem. It is important to note that (n,l) and m_{ready} are interdependent. Decreasing m_{ready} saves more energy, but it also leads to an increase in both n and l, resulting in an undesirable reduction in QoS, and vice versa.

Let $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{C})$ be an MDP where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of actions, $\mathcal{P}$ is the transition probability, and $\mathcal{C}$ the immediate cost of each action. Let $\mathcal{H}_A = (S_A, P_A)$ be the histogram used to model the arrival jobs, and let $\mathcal{H}_D = (S_D, P_D)$ be the histogram used to model the service rate. The state of the system is defined by the couple $((m_{stop}, m_{lat_k}, m_{lat_{k-1}}, \ldots, m_{lat_i}, \ldots, m_{lat_2}, m_{lat_1}, m_{ready}), (n,l))$ indeed the state space $\mathcal{S}$ is defined as:

$$\mathcal{S} = \{((m_x : x \in \mathcal{L}), (n,l)) \mid m_x \in [0..max],\, n \in [0..b] \text{ and } l \in [0..\max(S_A)]\} \quad (4)$$

The action space is defined as:

$$\mathcal{A} = \{\alpha_{+z} \mid z \in [1..max]\} \cup \{\alpha_0\} \cup \{\alpha_{-z} \mid z \in [1..max]\} \quad (5)$$

where:

1. action α_0 consists of doing nothing;
2. action α_{+z} consists in switching-on z additional servers if the number of stopped servers is more than z, otherwise switches-on all stopped machines;
3. action α_{-z} consists in switching-off z servers if the number of ready servers is more than z, otherwise switches-off all the ready machines.

If the current state of the system is:

$$s = \begin{cases} (m_{stop}, m_{lat_k}, m_{lat_{k-1}}, \ldots, m_{lat_i}, \ldots, m_{lat_2}, m_{lat_1}, m_{ready}) \\ (n,l) \end{cases} \quad (6)$$

then:

1. action α_0 move the system to the state s^0:

$$s^0 = \begin{cases} (m_{stop}, 0, m_{lat_k}, m_{lat_{k-1}}, \ldots, m_{lat_i}, \ldots, m_{lat_2}, m_{lat_1} + m_{ready}) \\ (n',l') \end{cases} \quad (7)$$

So we have a probability of $\mathcal{P}_{ss^0}^{\alpha_0}$ and an immediate cost of $\mathcal{C}_s^{\alpha_0}$ to transit from state s to state s^0 under action α_0:

$$\mathcal{P}_{ss^0}^{\alpha_0} = \sum_{\left\{\begin{array}{ll} \text{for} & \text{each } i \in S_A \text{ and each } d \in S_D \text{ satisfying:} \\ n' & = \min\{b, \max\{0, n+i-(m_{ready}+m_{lat_1}) \times d\}\} \\ l' & = \max\{0, n+a-(m_{ready}+m_{lat_1}) \times d - b\} \end{array}\right.} P_A(i) \times P_D(d) \quad (8)$$

$$\mathcal{C}_s^{\alpha_0} = n \times c_N + l \times c_L + (m_{lat_1} + m_{ready}) \times c_M + \textstyle\sum_{i=2}^{k} m_{lat_i} \times \frac{c_{on}}{k} \quad (9)$$

2. action α_{+z} move the system to the state s^+:

$$s^+ = \begin{cases} (\max\{m_{stop} - z, 0\}, \min\{z, m_{stop}\}, m_{lat_k}, m_{lat_{k-1}}, \ldots \\ \quad \ldots, m_{lat_i}, \ldots, m_{lat_2}, m_{lat_1} + m_{ready}) \\ (n',l') \end{cases} \quad (10)$$

So we have a probability of $\mathcal{P}_{ss^+}^{\alpha_{+z}}$ to transit from state s to state s^+ under action α_{+z}:

$$\mathcal{P}_{ss^+}^{\alpha_{+z}} = \sum_{\left\{\begin{array}{ll} \text{for} & \text{each } i \in S_A \text{ and each } d \in S_D \text{ satisfying:} \\ n' & = \min\{b, \max\{0, n+i-(m_{ready}+m_{lat_1}) \times d\}\} \\ l' & = \max\{0, n+a-(m_{ready}+m_{lat_1}) \times d - b\} \end{array}\right.} P_A(i) \times P_D(d) \quad (11)$$

And we have an immediate cost of $\mathcal{C}_s^{\alpha_{+z}}$ to transit from state s to state s^+ under action α_{+z}:

$$\begin{aligned} \mathcal{C}_s^{\alpha_{+z}} &= n \times c_N + l \times c_L + (m_{lat_1} + m_{ready}) \times c_M \\ &\quad + \Big(\min\{z, m_{stop}\} + \textstyle\sum_{i=2}^{k} m_{lat_i}\Big) \times \frac{c_{on}}{k} \end{aligned} \quad (12)$$

3. action α_{-z} move the system to the state s^-:

$$s^- = \begin{cases} (m_{stop} + \min\{z, m_{ready}\}, 0, m_{lat_k}, m_{lat_{k-1}}, \ldots \\ \quad \ldots, m_{lat_i}, \ldots, m_{lat_2}, m_{lat_1} + \max\{m_{ready} - z, 0\}) \\ (n',l') \end{cases} \quad (13)$$

So we have a probability of $\mathcal{P}_{ss^-}^{\alpha_{-z}}$ to transit from state s to state s^- under action α_{-z}:

$$\mathcal{P}_{ss^-}^{\alpha_{-z}} = \sum_{\left\{\begin{array}{ll} \text{for} & \text{each } i \in S_A \text{ and each } d \in S_D \text{ satisfying:} \\ n' & = \min\{b, \max\{0, n+i-(m_{lat_1}+\max\{m_{ready}-z,0\}) \times d\}\} \\ l' & = \max\{0, n+a-(m_{lat_1}+\max\{m_{ready}-z,0\}) \times d - b\} \end{array}\right.} P_A(i) \times P_D(d) \quad (14)$$

And we have an immediate cost of $\mathcal{C}_s^{\alpha_{-z}}$ to transit from state s to state s^- under action α_{-z}:

$$\begin{aligned}\mathcal{C}_s^{\alpha_{-z}} &= n \times c_N + l \times c_L + (m_{lat_1} + \max\{m_{ready} - z, 0\}) \times c_M \\ &\quad + \sum_{i=2}^{k} m_{lat_i} \times \frac{c_{on}}{k}\end{aligned} \tag{15}$$

Notice that n' (resp. l') is the new number of waiting (resp. rejected) jobs, and the immediate cost includes two parts. The QoS part, which includes cost due to waiting and rejected jobs. Power part composed of energy consumed for running operational servers and energy used by servers which are in the switching-on latency period. Table 2 resumes parameters of MDP model formulation.

Table 2: Model and MDP Parameters.

Parameters	Description
h	duration of analysis
k	latency period
b	buffer size
max	total number of servers
$\mathcal{H}_D$	histogram modeling the processing capacity of a server
$\mathcal{H}_A$	histogram of job arrivals
m_{ready}	number of operational servers
m_{lat_i}	number of servers in latency level i
n	number of waiting jobs
l	number of rejected jobs
$\mathcal{S}$	set of all possible states
$\mathcal{A}$	set of all possible actions
s	$s = ((m_x : x \in \mathcal{L}), (n,l))$ system state
s_0	$s_0 = ((max, 0, ..., 0), (0,0))$ starting state
α_0	action to keep the same number of operational servers
α_{+z}	action to switch-on z additional server
α_{-z}	action to switch-off z server
$\mathcal{P}_{ss'}^{\alpha}$	probability of transition from s to s' under action α
$\mathcal{C}_s^{\alpha}$	immediate cost from s under action α

Theorem 1. *The number of state of the MDP is in* $O\Big(b \times \max(S_A) \times max^k\Big)$.

Proof. Every state of the MDP includes three parts: (i) the number of waiting jobs in the buffer which is bounded by b, (ii) the number of rejected jobs which can be at most equals to the maximum number of arrival jobs given by $\max(S_A)$, and (iii) as we deal with k level of latency, the set of number of servers for every latency level in $\mathcal{L}$. Each number is between 0 and max. So, $|\mathcal{S}|$ is bounded by $(b+1) \times (\max(S_A)+1) \times max^{k+2}$. □

Theorem 2. *MDP transitions number is in* $O\Big(b \times |S_A| \times |S_D| \times \max(S_A) \times max^k\Big)$.

Proof. From each state of the MDP we have at most $(2 \times max + 1)$ action, and each action leads to a number of transitions equals to $|S_A| \times |S_D|$ (one transition for each bin in the support of the arrivals distribution combined with each bin in the support of the service rate distribution). □

To illustrate our formalization, Figure 3 shows an example of an MDP modeling a very simple data center of two servers $max = 2$ with a latency period of two unite of time $k = 2$ with a buffer size equals two $b = 2$. Additionally, to keep the example simple, we set service rate of each machine as a histogram with only one bin: $P_D(1) = 1$ (one server process one job every slot). Also job arrivals are modeled by histogram with only one bin $P_A(1) = 1$ (system receives one job every slot).

7 OPTIMAL STRATEGY

By formulating our optimization problem as a Markov Decision Process (MDP), we define an action as the process of turning on a specific number of servers each unit of time and turning off the rest of the servers. The optimal strategy is determined by finding the best sequence of actions that minimizes the overall cost during a finite period of time, referred to as the horizon and denoted by h. In general, the objective of the value function $V : \mathcal{S} \times [0..h] \rightarrow$ I!R$^+$ is to minimize the expected sum of costs over time:

$$V(s,t) = \min_{\pi} \mathbb{E}\left[\sum_{k=0}^{t} \mathcal{C}_{s_k}^{\pi(s_k,k)}\right] \tag{16}$$

The value function can be seen as a Bellman equation (Bellman, 1957):

$$V(s,t) = \min_{\alpha}\left\{\mathcal{C}_s^{\alpha} + \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{\alpha} V(s', t-1)\right\} \tag{17}$$

Where α is the action taken by the system, and $\mathcal{P}_{ss'}^{\alpha}$ is the transition probability from state s to state s'. In this case the optimal policy for each state s is:

$$\pi^*(s,t) = \operatorname*{argmin}_{\alpha \in \mathcal{A}} \left\{ \mathcal{C}_s^\alpha + \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^\alpha V(s', t-1) \right\} \quad (18)$$

Given that the value iteration algorithm is a powerful dynamic programming approach for solving the Bellman equation, we employed it to derive the optimal control policy of our Markov Decision Process (MDP).

8 CASE STUDY

This case study uses real traffic traces based on the open *cluster-data-2011-2* trace (Wilkes, 2011). As in (Bayati, 2018), we model arrival jobs and service rate by discrete distributions build from the job/machine events corresponding to the requests destined to a specific Google data center for the whole month of May 2011. This traffic trace is sampled with a sampling period of 136 second that ensure the i.i.d-ness.

Listing 1: Example of PRISM specification.

```
mdp
const int k=2;
const int B;
const int m=5;
const int d=2;
//----------------------------------------------
const double cN =1;
const double cM =1;
const double cL =1;
const double cON=1;
//----------------------------------------------
const int    A0=7;
const double p0=1.0;
//----------------------------------------------
module system1
off   : [0..m ] init m;
set2  : [0..m ] init 0;
set1  : [0..m ] init 0;
M     : [0..m ] init 0;
N     : [0..B ] init 0;
L     : [0..A0] init 0;
//----------------------------------------------
 [a_5] M>=5
 -> p0:(N'=min(B,max(0,N+A0-M*d)))&
 (M'=min(m,M+set1-5))&(set1'=set2)&(set2'=0)&
 (off'=min(m,off+5))&(L'=max(0,N+A0-M*d-B));
//----------------------------------------------
 [a_4] M>=4
 -> p0:(N'=min(B,max(0,N+A0-M*d)))&
 (M'=min(m,M+set1-4))&(set1'=set2)&(set2'=0)&
 (off'=min(m,off+4))&(L'=max(0,N+A0-M*d-B));
//----------------------------------------------
 [a_3] M>=3
 -> p0:(N'=min(B,max(0,N+A0-M*d)))&
 (M'=min(m,M+set1-3))&(set1'=set2)&(set2'=0)&
 (off'=min(m,off+3))&(L'=max(0,N+A0-M*d-B));
```

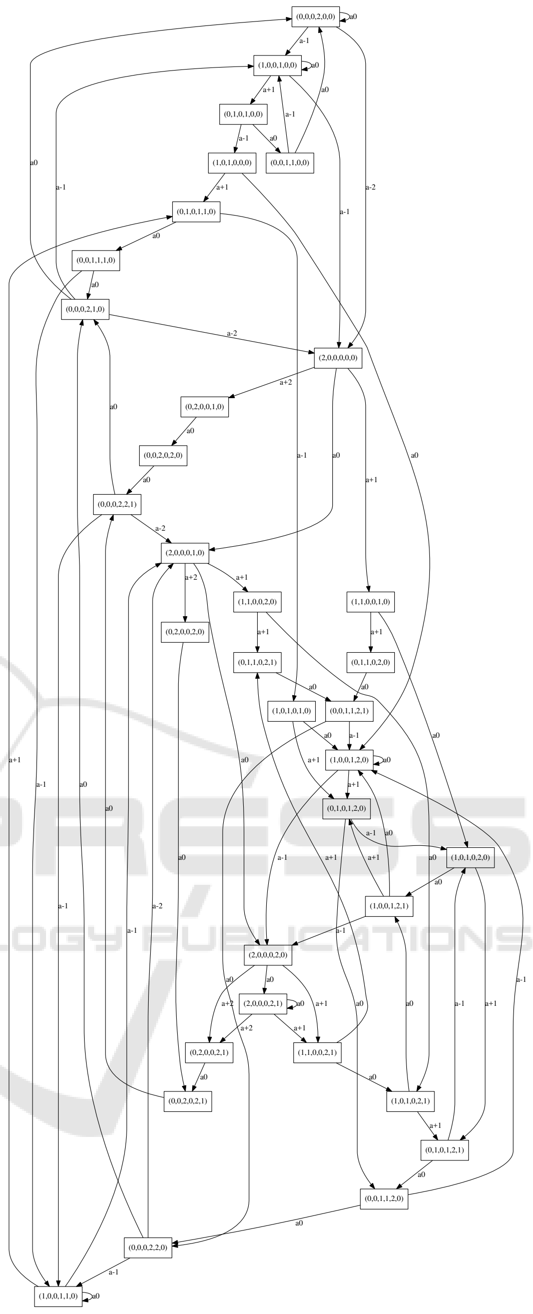

Figure 3: MDP example for a data center of only two machine with latency $k = 2$ and $b = 2$.

```
//----------------------------------------------
 [a_2] M>=2
 -> p0:(N'=min(B,max(0,N+A0-M*d)))&
 (M'=min(m,M+set1-2))&(set1'=set2)&(set2'=0)&
 (off'=min(m,off+2))&(L'=max(0,N+A0-M*d-B));
//----------------------------------------------
 [a_1] M>=1
 -> p0:(N'=min(B,max(0,N+A0-M*d)))&
 (M'=min(m,M+set1-1))&(set1'=set2)&(set2'=0)&
 (off'=min(m,off+1))&(L'=max(0,N+A0-M*d-B));
```

```
//---------------------------------------------
[a0] off>=0
-> p0:(N'=min(B,max(0,N+A0-M*d)))&
(M'=min(m,M+set1))&(set1'=set2)&(set2'=0)&
(off'=off-0)&(L'=max(0,N+A0-M*d-B));
//---------------------------------------------
[a1] off>=1
-> p0:(N'=min(B,max(0,N+A0-M*d)))&
(M'=min(m,M+set1))&(set1'=set2)&(set2'=1)&
(off'=off-1)&(L'=max(0,N+A0-M*d-B));
//---------------------------------------------
[a2] off>=2
-> p0:(N'=min(B,max(0,N+A0-M*d)))&
(M'=min(m,M+set1))&(set1'=set2)&(set2'=2)&
(off'=off-2)&(L'=max(0,N+A0-M*d-B));
//---------------------------------------------
[a3] off>=3
-> p0:(N'=min(B,max(0,N+A0-M*d)))&
(M'=min(m,M+set1))&(set1'=set2)&(set2'=3)&
(off'=off-3)&(L'=max(0,N+A0-M*d-B));
//---------------------------------------------
[a4] off>=4
-> p0:(N'=min(B,max(0,N+A0-M*d)))&
(M'=min(m,M+set1))&(set1'=set2)&(set2'=4)&
(off'=off-4)&(L'=max(0,N+A0-M*d-B));
//---------------------------------------------
[a5] off>=5
-> p0:(N'=min(B,max(0,N+A0-M*d)))&
(M'=min(m,M+set1))&(set1'=set2)&(set2'=5)&
(off'=off-5)&(L'=max(0,N+A0-M*d-B));
endmodule
//---------------------------------------------
rewards "r"
[a_5]  true : M*cM+N*cN+L*cL+(set1+set2)*cON/k;
[a_4]  true : M*cM+N*cN+L*cL+(set1+set2)*cON/k;
[a_3]  true : M*cM+N*cN+L*cL+(set1+set2)*cON/k;
[a_2]  true : M*cM+N*cN+L*cL+(set1+set2)*cON/k;
[a_1]  true : M*cM+N*cN+L*cL+(set1+set2)*cON/k;
[a0]   true : M*cM+N*cN+L*cL+(set1+set2)*cON/k;
[a1]   true : M*cM+N*cN+L*cL+(set1+set2)*cON/k;
[a2]   true : M*cM+N*cN+L*cL+(set1+set2)*cON/k;
[a3]   true : M*cM+N*cN+L*cL+(set1+set2)*cON/k;
[a4]   true : M*cM+N*cN+L*cL+(set1+set2)*cON/k;
[a5]   true : M*cM+N*cN+L*cL+(set1+set2)*cON/k;
endrewards
```

To specify, solve, and analyze energy consumption in our data centers, we utilize an efficient model checker called PRISM (Kwiatkowska et al., 2002). PRISM is a probabilistic software that allows for the specification of MDP models. However, writing a specification for large-scale systems can be challenging. For instance, the PRISM program for a data center with a latency of ten units of time consists of thousands of lines of code, and it is difficult to write the specification manually without making mistakes or omitting cases. Furthermore, updating or maintaining such a large specification is time-consuming. Therefore, automated generation of the specification is necessary to save time and avoid errors. (see Theorems 1 and 2), We developed a tool to automatically generate the PRISM specification file. For instance, consider a basic data center with five machines and latency $k = 2$, and Dirac histograms for job arrival and service rate. The corresponding PRISM script for this example is shown in Listing 1.

Table 3: Parameters of the numerical analysis.

b	max	$\mathbb{E}(\mathcal{H}_A)$	$\mathbb{E}(\mathcal{H}_D)$	c_M	c_{On}	c_N	c_L	k	h
1-70	5	7	2	1	1	1	1	2-5	100

Figure 4 shows the result of experiments where we analyze the total cost over 100 units of time when varying the buffer size b and latency k. We observe that in general the total cost increase when b is less than some value b_k that depends on k.

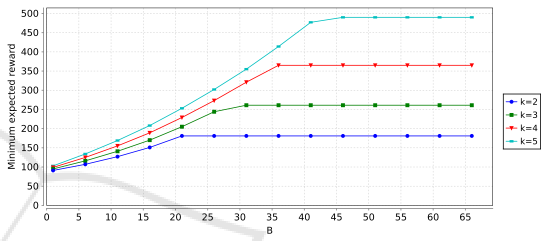

Figure 4: Total cost when varying buffer size b for different level k.

When the buffer size b is greater than b_k, the total cost appears to converge for any configuration. This behavior can be explained as follows: when the buffer size is small, the number of waiting jobs is low, resulting in a small number of served jobs, and the system switches on fewer servers. As a result, both the energy consumption and the number of waiting jobs are low. However, when the buffer size increases, the number of waiting jobs becomes more significant, resulting in a larger number of served jobs and more servers being switched on. This, in turn, leads to an increase in both energy consumption and the number of waiting jobs. As the number of incoming jobs is limited, when the buffer size exceeds a certain value, the number of waiting jobs and the number of served jobs become stable, resulting in a constant number of running servers and, as a result, a constant total cost. Another related observation is the fact that b_k grows when k increases. For example, for a system with $k = 4$, the total cost becomes independent from the latency period when $b > b_k = b_4 = 36$, however, for a system with $k = 2$, the total cost becomes independent from the latency period when only $b > b_k = b_2 = 21$. It means that a data center with servers with longer periods of latency have to be designed with more longer buffer size. A last observation is the fact that the total cost is more important when k increases. We can explain that from the fact that the system, during the latency period, accumulates more waiting jobs before the complete switching-on of the servers.

9 CONCLUSION

The aim of this study is to develop a model for efficient management of a data center that takes into account server latency and minimizes energy consumption and Quality of Service (QoS) costs. The model uses a discrete-time Markov decision process, with job arrivals and service rates modeled by a discrete probability distribution estimated from real data. To account for switching-on latency, it is assumed that all servers have the same constant latency period of k units of time. The optimal control policy is computed using the value iteration algorithm, and is used to implement a Dynamic Power Management strategy that balances energy consumption and performance. Despite the large size of the model (which is an exponential of k), the experimental and theoretical results demonstrate that increasing buffer size can lead to significant energy savings when the latency is higher.

REFERENCES

Aidarov, K., Ezhilchelvan, P. D., and Mitrani, I. (2013). Energy-aware management of customer streams. *Electr. Notes Theor. Comput. Sci.*, 296:199–210.

An, H. and Ma, X. (2022). Dynamic coupling real-time energy consumption modeling for data centers. *Energy Reports*, 8:1184–1192.

Bayati, M. (2018). Power management policy for heterogeneous data center based on histogram and discrete-time mdp. *Electronic Notes in Theoretical Computer Science*, 337:5 – 22. Proceedings of the Ninth International Workshop on the Practical Application of Stochastic Modelling (PASM).

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition.

Benini, L., Bogliolo, A., Paleologo, G. A., and De Micheli, G. (1999). Policy optimization for dynamic power management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(6):813–833.

Benson, T., Akella, A., and Maltz, D. A. (2010). Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 267–280. ACM.

Chen, G. and Wang, X. (2022). Performance optimization of machine learning inference under latency and server power constraints. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 325–335. IEEE.

Dyachuk, D. and Mazzucco, M. (2010). On allocation policies for power and performance. In *2010 11th IEEE/ACM International Conference on Grid Computing*, pages 313–320. IEEE.

Entezari-Maleki, R., Sousa, L., and Movaghar, A. (2017). Performance and power modeling and evaluation of virtualized servers in iaas clouds. *Information Sciences*, 394:106–122.

Gandhi, A., Doroudi, S., Harchol-Balter, M., and Scheller-Wolf, A. (2013). Exact analysis of the m/m/k/setup class of markov chains via recursive renewal reward. In *ACM SIGMETRICS Performance Evaluation Review*, volume 41, pages 153–166. ACM.

Greenberg, A. G., Hamilton, J. R., Maltz, D. A., and Patel, P. (2009). The cost of a cloud: research problems in data center networks. *Computer Communication Review*, 39(1):68–73.

Kendall, M. (1973). *Time-series*. Griffin.

Khan, W. (2022). Advanced data analytics modelling for evidence-based data center energy management.

Kwiatkowska, M., Norman, G., and Parker, D. (2002). Prism: Probabilistic symbolic model checker. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 200–204. Springer.

Lee, Y. C. and Zomaya, A. Y. (2012). Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280.

Leng, B., Krishnamachari, B., Guo, X., and Niu, Z. (2016). Optimal operation of a green server with bursty traffic. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

Maccio, V. J. and Down, D. G. (2015). On optimal control for energy-aware queueing systems. In *Teletraffic Congress (ITC 27), 2015 27th International*, pages 98–106. IEEE.

Mitrani, I. (2013). Managing performance and power consumption in a server farm. *Annals OR*, 202(1):121–134.

Peng, X., Bhattacharya, T., Mao, J., Cao, T., Jiang, C., and Qin, X. (2022). Energy-efficient management of data centers using a renewable-aware scheduler. In *2022 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pages 1–8. IEEE.

Rajesh, C., Dan, S., Steve, S., and Joe, T. (2008). Profiling energy usage for efficient consumption. *The Architecture Journal*, 18:24.

Rteil, N., Burdett, K., Clement, S., Wynne, A., and Kenny, R. (2022). Balancing power and performance: A multi-generational analysis of enterprise server bios profiles. In *2022 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*, pages 81–85. IEEE.

Schwartz, C., Pries, R., and Tran-Gia, P. (2012). A queuing analysis of an energy-saving mechanism in data centers. In *Information Networking (ICOIN), 2012 International Conference on*, pages 70–75.

Wilkes, J. (2011). More Google cluster data. Google research blog. Posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.

Xu, X. and Tian, N. (2008). The m/m/c queue with (e, d) setup time. *Journal of Systems Science and Complexity*, 21(3):446–455.