Reinforcement Learning-based Adaptive Resource Management of Differentiated Services in Geo-distributed Data Centers

Xiaojie Zhou[†], Kun Wang[†‡], Weijia Jia[†*] and Minyi Guo[†]
[†]Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, P. R. China, 200240
Email: szxjzhou@sjtu.edu.cn, jia-wj@cs.sjtu.edu.cn, guo-my@cs.sjtu.edu.cn
[‡]Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks
Nanjing University of Posts and Telecommunications, Nanjing, P. R. China, 210042
Email: kwang@njupt.edu.cn

***Abstract*—For better service provision and utilization of renewable energy, Internet service providers have already built their data centers in geographically distributed locations. These companies balance quality of service (QoS) revenue and power consumption by migrating virtual machines (VMs) and allocating the resource of servers adaptively. However, existing approaches model the QoS revenue by service-level agreement (SLA) violation, and ignore the network communication cost and immigration time. In this paper, we propose a reinforcement learning-based adaptive resource management algorithm, which aims to get the balance between QoS revenue and power consumption. Our algorithm does not need to assume prior distribution of resource requirements, and is robust in actual workload. It outperforms other existing approaches in three aspects: 1) The QoS revenue is directly modeled by differentiated revenue of different tasks, instead of using SLA violation. 2) For geo-distributed data centers, the time spent on VM migration and network communication cost are taken into consideration. 3) The information storage and random action selection of reinforcement learning algorithms are optimized for rapid decision making. Experiments show that our proposed algorithm is more robust than the existing algorithms. Besides, the power consumption of our algorithm is around 13.3% and 9.6% better than the existing algorithms in non-differentiated and differentiated services.**

***Index Terms*—Geo-distributed data centers; Differentiated services; QoS revenue; Power consumption; Reinforcement learning**

I. Introduction

With the development of cloud computing, Microsoft, Google, Amazon and other companies have already built their data centers and provided their own cloud services to the public. For better service provision and use of renewable energy, data centers are generally deployed in geographically distributed locations [1]. More and more tasks with different quality of service (QoS) requirements have been submitted to the data centers for processing. The QoS directly affects user's satisfaction, and service provider's business future [2]. Meanwhile, the power consumption and CO_2 emissions of data centers are increasing with the rapid rise of the amount of tasks [3], [4].

In order to find the balance between QoS revenue and power consumption, varies of models have been proposed. However, adaptive resource management of differentiated services in geo-distributed data centers has been so far addressed in isolation. In the area of adaptive resource management, recent studies have focused primarily on assuming prior distribution of resource requirements [5], [6] or learning resource requirements based on history utilization [7], [8]. For the former, as the network environment is unstable in reality, it seems impossible for us to assume an accurate prior distribution of resource requirements [7]. For the latter, they formulate the balance between QoS revenue and power consumption and propose corresponding heuristic algorithms. However, these algorithms are suffered from curse of dimension in solving large-scale Markov decision process (MDP) problems [7]. Besides, they mainly model QoS revenue by service-level agreement (SLA) violation or other similar ways [7], [8]. Under SLA violation, any task gets zero revenue when no violation happens and has the same linear gain loss of delay. Moreover, most of them ignore the time spent on VM migration and network communication cost, which are important issues in geo-distributed data centers.

For the low latency characteristic of data centers and the instability of the actual network environment, the low-complexity and high-robustness should be taken into account in algorithm design. Besides, the first-order transition probability of the VMs' resource demands is also quasi-static for a long period and non-uniformly distributed by properly choosing the time-slot duration [7]. Therefore, it is suitable for us to adopt reinforcement learning algorithms in finding such balance, which are algorithms with low computational complexity based on Markov chain model [9]. Among varieties of reinforcement learning algorithms, Q-learning algorithm [10]

[*] Corresponding author

has the advantage in rapid decision making. However, the huge size of state and action set, and the random action selection strategy of traditional Q-learning algorithm makes traditional algorithm infeasible to solve large-scale MDP problems [7]. Therefore, we have to reduce the size of the sets and optimize random action selection.

In this paper, we regard resource management as a stochastic optimization problem. We utilize the Markov property of differentiated QoS revenue and power consumption, and as a result, adopt the Markov chain in modeling. Our contributions can be summarized as follows:

- We formulate the resource allocation problem of differentiated services in geo-distributed data centers as an infinite horizon MDP problem with an objective to finding tradeoff of QoS revenue and power consumption. Different from the existing algorithms, the QoS revenue is directly modeled by differentiated revenue of different tasks, instead of using SLA violation. Besides, the time spent on VM migration and network communication cost are also taken into consideration for geo-distributed data centers.
- We propose a reinforcement learning based adaptive resource management algorithm in solving such problem, which fits with geo-distributed data centers for low-complexity and high-robustness requirements. Moreover, information storage and random action selection are optimized for large-scale MDP problems. Besides, the convergence of our algorithm is also analyzed.
- We conduct detailed simulations in CloudSim [11] based on real world workload trace. The experimental results show that the average power consumption of our algorithm can be reduced by up to 13.3% and 9.6% when compared with the existing algorithms in non-differentiated and differentiated services, respectively.

The remainder of this paper is organized as follows. In Section II, we model the geo-distributed data centers, formulate and analyze the resource allocation problem of differentiated services. Then, our algorithm is described in Section III. The experimental setup and experimental results of our algorithm are described in Section IV. Our conclusion is in Section V.

II. Modeling and Problem Formulation

A. System Model

We consider a three-layer geo-distributed data centers with servers $S = \{s_1, ..., s_M\}$ and VMs $V = \{v_1, ..., v_N\}$, which is shown in Fig. 1. The location, resource constraints, resource required, and resource allocated of s_i at time t are denoted as $l_i, c_i, r_i(t)$, and $a_i(t)$, respectively. Besides, we let $\mathfrak{mi}_i, \mathfrak{p}_i(t), \mathfrak{pr}_i(t), \mathfrak{l}_i(t), \mathfrak{r}_i(t)$, and $\mathfrak{a}_i(t)$ denote million instructions (MI), QoS revenue (when finished at t), progress rate, location, resource required, and resource allocated of the task in v_i at t, respectively. The data flow between these three layers are transmitted by core switch, aggregation switches and top-of-rack (ToR) switches, respectively [12].

All VMs located in $s_i \in S$ share the physical resources of this server. Virtual machine monitor (VMM) dynamically

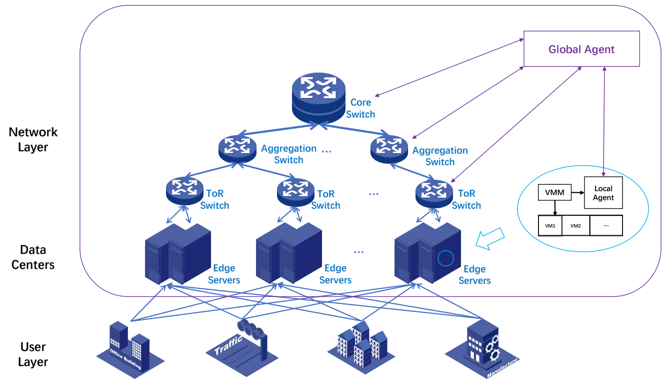


Fig. 1. Typical geo-distributed data centers architecture

monitors $\mathfrak{r}_j(t)$ of v_j located in s_i, and determines how resources are allocated. If $r_i(t) > c_i$, $\mathfrak{a}_j(t) < \mathfrak{r}_j(t)$, resulting in a performance decrease.

B. Problem Formulation

Our target is to find the best balance between QoS revenue $\mathcal{R}_{QoS}(t)$ and power consumption $\mathcal{R}_{Power}(t)$ until t. Therefore, total revenue $\mathcal{R}(t)$ is defined as

$$\mathcal{R}(t) = \mathcal{R}_{QoS}(t) + k\mathcal{R}_{Power}(t),$$

where k is the weight of power consumption.

As for $\mathcal{R}_{QoS}(t)$, existing algorithms mainly model it by SLA violation [8]. However, SLA violation of each VM is the same in SLA violation modeling. This is not consistent with the fact that different tasks have different QoS requirements. Therefore, we directly model $\mathcal{R}_{QoS}(t)$ by the profit of each task at different finishing time, which is shown as

$$\mathcal{R}_{QoS}(t) = \int \sum_{i=1}^{N} \mathfrak{p}_i(t) 1\{\mathfrak{pr}_i(t) = 1\} dt. \tag{1}$$

For s_i, if $a_i(t) > 0$, its power consumption $\mathcal{R}_{Power}(t)$ is proportional to resource utilization as [13]

$$\mathcal{R}_{Power}(t) = -\int \sum_{i=1}^{M} \left[p_{e_i} + (p_{m_i} - p_{e_i}) \frac{a_i(t)}{c_i} \right] dt, \tag{2}$$

where p_{e_i} and p_{m_i} represent the power consumption of 0% and 100% CPU utilization of s_i, respectively. Otherwise, $s_i \approx 0$ [7].

Therefore, our problem is defined as follows:

Problem 1. *The adaptive resource management allocation problem of differentiated services in geo-distributed data centers:*

$$\begin{aligned} \max \quad & \mathcal{R}(t) = \mathcal{R}_{QoS}(t) + k\mathcal{R}_{Power}(t) \\ s.t. \quad & a_i(t) \le c_i, && i = 1, 2, ..., M \\ & a_i(t) = \textstyle\sum_{j=1}^{N} 1\{\mathfrak{l}_j(t) = i\} \mathfrak{a}_j(t) && i = 1, 2, ..., M \end{aligned},$$

C. Problem Analysis

In this problem, both $\mathcal{R}_{QoS}(t)$ and $\mathcal{R}_{Power}(t)$ can be represented as the addition of $\bar{\mathcal{R}}_{QoS_i}(T_j)$ of v_i and $\bar{\mathcal{R}}_{Power_i}(T_j)$ of s_i during j^{th} time-slot T_j, which are shown as

$$\mathcal{R}_{QoS}(T_k) = \sum_{i=1}^{N}\sum_{j=1}^{k}\bar{\mathcal{R}}_{QoS_i}(T_j), \tag{3}$$

$$\bar{\mathcal{R}}_{QoS_i}(T_j) = \int_{t_0+(j-1)T}^{t_0+jT} \mathfrak{p}_i(t)1\{\mathfrak{pr}_i(t)=1\}dt, \tag{4}$$

$$\mathcal{R}_{Power}(T_k) = \sum_{i=1}^{M}\sum_{j=1}^{k}\bar{\mathcal{R}}_{Power_i}(T_j), \tag{5}$$

and

$$\bar{\mathcal{R}}_{Power_i}(T_j) = -\int_{t_0+(j-1)T}^{t_0+jT}\left[p_{e_i} + (p_{m_i}-p_{e_i})\frac{a_i(t)}{c_i}\right]dt, \tag{6}$$

where t_0 is the start time of the system, and T is the length of each time-slot. Therefore, $\mathcal{R}_{QoS}(T_k)$ and $\mathcal{R}_{Power}(T_k)$ are shown as

$$\begin{aligned}\mathcal{R}_{QoS}(T_k) &= \bar{\mathcal{R}}_{QoS}(T_k) + \mathcal{R}_{QoS}(T_{k-1}),\\ \mathcal{R}_{Power}(T_k) &= \bar{\mathcal{R}}_{Power}(T_k) + \mathcal{R}_{Power}(T_{k-1}).\end{aligned}$$

We adopt Q-learning algorithm [10] in solving this problem. The learning rule is shown as

$$\begin{aligned}\mathcal{Q}(\mathcal{S}_{j-1},\mathcal{A}_{j-1}) &= (1-\alpha)\mathcal{Q}(\mathcal{S}_{j-1},\mathcal{A}_{j-1})\\ &+\alpha\left[\boldsymbol{r}_{j-1} + \gamma\max_{\mathcal{A}_j}\mathcal{Q}(\mathcal{S}_j,\mathcal{A}_j)\right],\end{aligned} \tag{7}$$

where $\mathcal{S}_j, \mathcal{A}_j, \boldsymbol{r}_j, \alpha$, and γ represent the state, action, reward function, learning rate, and discount parameter of T_j, respectively.

III. Reinforcement Learning Based Adaptive Resource Management

A. Q-learning Settings

From Eqs. (1), (2), we can conclude the state set $\mathbb{S}$ as $\mathbb{S} = \{\mathfrak{Pr} = \{\mathfrak{pr}_1,...,\mathfrak{pr}_N\}, \mathfrak{L} = \{\mathfrak{l}_1,...,\mathfrak{l}_N\}, \mathfrak{R} = \{\mathfrak{r}_1,...,\mathfrak{r}_N\}\}$. Besides, the action set $\mathbb{A} = \{\mathfrak{L}\} = \{\mathfrak{l}_1,...,\mathfrak{l}_N\}$. The reward function $\boldsymbol{r}_j$ contains the reward of power consumption $\boldsymbol{r}_{power_i}(T_j)$ of s_i and QoS revenue $\boldsymbol{r}_{QoS_i}(T_j)$ of v_i. $\boldsymbol{r}_{power_i}(T_j)$ can be calculated by Eq. (6). However, for $\boldsymbol{r}_{QoS_i}(T_j)$, Eq. (4) only gets the revenue when task is finished, which can not reflect the progress of the task in time. Instead, we define $\boldsymbol{r}_{QoS_i}(T_j)$ based on estimated QoS revenue $\hat{\mathfrak{p}}_i(T_j)$ of v_i after T_j, which is defined as

$$\hat{\mathfrak{p}}_i(T_j) = \begin{cases}\mathfrak{p}_i(t_{fin_i}), & t_0+(j-1)T < t_{fin_i} \le t_0+jT\\ \mathfrak{p}_i\left(t_0+jT+\frac{1-\mathfrak{pr}_i(t_0+jT)}{\frac{\mathfrak{pr}_i(t_0+jT)}{jT-t_{start_i}}}\right), & \\ & 0<\mathfrak{pr}_i(t_0+jT)<1\\ 0, & others\end{cases},$$

where t_{start_i}, t_{fin_i} are the start time and finishing time of v_i, respectively. As a result, $\boldsymbol{r}_{QoS_i}(T_j)$ is obtained as

$$\begin{aligned}\boldsymbol{r}_{QoS_i}(T_j) &= \mathfrak{pr}_i(t_0+jT)\times\hat{\mathfrak{p}}_i(T_j)\\ &- \mathfrak{pr}_i(t_0+(j-1)T)\times\hat{\mathfrak{p}}_i(T_{j-1}).\end{aligned} \tag{8}$$

Therefore,

$$\boldsymbol{r}_i = \sum_{j=1}^{N}\boldsymbol{r}_{QoS_j}(T_i) - k\sum_{j=1}^{M}\boldsymbol{r}_{Power_j}(T_i).$$

where $\boldsymbol{r}_{QoS_j}(T_j), \boldsymbol{r}_{Power_j}(T_i)$ are obtained as Eqs. (8) and (6), respectively.

In this problem, it seems impossible for us to handle such a huge Q-matrix in large-scale MDP problems. Therefore, the optimization of Q-matrix storage and calculation is the first issue. Meanwhile, the system will sometimes chooses a very bad action, so the second issue is how to optimize random action selection. The analysis and corresponding optimization methods of these two issues are proposed in the next part.

B. Algorithm Optimization

1) ***Optimization of Q-matrix:*** In here, Q-matrix can be transformed to $\Omega = \{\omega_1,...,\omega_k\}, k \le \mathcal{M}$. ω_i stores a state $\mathbf{S}_i$, a set of corresponding actions $\mathbb{A}_i = \{\mathbf{A}_{i1},...,\mathbf{A}_{il}\}, l \le \mathcal{N}$ and Q-values $\mathbb{Q}_i = \{\mathbf{Q}_{i1},...,\mathbf{Q}_{il}\}, l \le \mathcal{N}$.

Since only part of the knowledge of Q-matrix is stored, allocation decision is made based on the similar state in Ω. For $\mathfrak{Pr}$ and $\mathfrak{R}$, Pearson correlation coefficient Γ and Euclidean distance $\mathcal{D}$ can be used to calculate the similarity.

We first set similarity threshold of $\mathfrak{R}$ and $\mathfrak{Pr}$ as $th_r = \{th_{\Gamma_r}, th_{\mathcal{D}_r}\}$ and $th_{pr} = \{th_{\Gamma_{pr}}, th_{\mathcal{D}_{pr}}\}$ in both Γ and $\mathcal{D}$. Then, for each new state $\mathcal{S}$, $\Gamma_r(\mathcal{S},\mathbf{S}_i), \Gamma_{pr}(\mathcal{S},\mathbf{S}_i), \mathcal{D}_r(\mathcal{S},\mathbf{S}_i)$, and $\mathcal{D}_{pr}(\mathcal{S},\mathbf{S}_i)$ $(i = \{1,...,k\})$ are calculated. The state $\mathcal{S}_i \in \Omega$ is called an alternative state, which lets $\Gamma_r(\mathcal{S},\mathbf{S}_i) \ge th_{\Gamma_r}, \Gamma_{pr}(\mathcal{S},\mathbf{S}_i) \ge th_{\Gamma_{pr}}, \mathcal{D}_r(\mathcal{S},\mathbf{S}_i) \le th_{\mathcal{D}_r}$ and $\mathcal{D}_{pr}(\mathcal{S},\mathbf{S}_i) \le th_{\mathcal{D}_{pr}}$.

All the alternative states are put into the set Θ. As massive migration of VMs would lead to severe network congestion, we set thresholds $th_{ToR}, th_{aggr}, th_{core}$ of each layer's transit data in a time-slot. The states with massive data transit are removed from Θ and at last the action in Θ with maximum Q-value is chosen. If $\Theta = \emptyset$ after removal, the action will be chosen randomly.

Meanwhile, $\mathbb{A}$ is also in non-polynomial size. For locations, $\mathcal{D}$ is used to measure the similarity. Similarly, threshold $th_{\mathcal{D}_{action}}$ is defined for the action. Then for each new action $\mathcal{A}$, we calculate $\mathcal{D}(\mathcal{A},\mathbf{A}_{ij}), j \le l$ for corresponding item ω_i. If $\exists m, \mathcal{D}(\mathcal{A},\mathbf{A}_{im}) \le \mathcal{D}(\mathcal{A},\mathbf{A}_{ij}) \le th_{\mathcal{D}_{action}}, j \le l$, we regard $\mathcal{A}$ as $\mathbf{A}_{im}$. Otherwise, $\mathcal{A}$ will be added to corresponding ω $(l < \mathcal{N})$, or replace the most similar one $(l = \mathcal{N})$.

2) ***Optimization of Random Action Selection:*** We analyze the random action selection of **under-utilized**, **normal-utilized** and **over-utilized** servers. For under-utilized or normal-utilized servers, our purpose is to reduce power consumption by migrating all the VMs to other non-empty servers, without any over-utilized servers addition. Meanwhile, for over-utilized servers, our purpose is to reduce utilization by migrating some VMs to other servers. First of all, we set over-utilized threshold th_{over} in advance and calculate estimated migration revenue $\hat{\boldsymbol{r}}_{mig_i}(T_j)$ of s_i during T_j.

Under-utilized or **normal-utilized** servers: $\hat{\boldsymbol{r}}_{mig_i}(T_j)$ is obtained as

$$\hat{\boldsymbol{r}}_{mig_i}(T_j) = k\boldsymbol{r}_{Power_i}(T_j) - \hat{\boldsymbol{r}}_{mt_i}(T_j) - k\hat{\boldsymbol{r}}_{pc_i}(T_j), \quad (9)$$

where $\hat{\boldsymbol{r}}_{mt_i}(T_j), \hat{\boldsymbol{r}}_{pc_i}(T_j)$ are the estimated QoS revenue loss and power consumption due to migration, respectively.

For $\hat{\boldsymbol{r}}_{mt_i}(T_j)$, the transmission delay and the propagation delay are taken into consideration. For the former, as the bottleneck of transmission is ToR switches, the bandwidth of ToR switches is used to calculate the transmission delay. For the latter, we estimate the destination of migrated VM by expected location. For s_i, weight $w_k (k \neq i)$ is assigned for s_k. If s_k is empty, $w_k = 0$. Otherwise, w_k is obtained as

$$w_k = \max\left\{0, th_{over} - \frac{a_k(t_0+jT) + \frac{a_i(t_0+jT)}{\#VM\ in\ s_i}}{c_k}\right\}.$$

Therefore, $\hat{\boldsymbol{r}}_{mt_i}(T_j)$ is obtained as

$$\hat{\boldsymbol{r}}_{mt_i}(T_j) = \frac{\left\| \sum_{k \neq i} \frac{w_k \times l_k}{w_k} - l_i \right\|}{3 \times 10^8\ m/s} + \frac{a_i(t_0+jT)}{bw_{ToR}}, \quad (10)$$

where bw_{ToR} is the bandwidth of ToR switch.

Similarly, we calculate $\hat{\boldsymbol{r}}_{pc_i}(T_j)$ as

$$\hat{\boldsymbol{r}}_{pc_i}(T_j) = \sum_{k \neq i} \frac{w_k \times (p_{m_k} - p_{e_k})}{w_k} \times \frac{a_i(t_0+jT)}{c_i}. \quad (11)$$

Over-utilized servers: $\hat{\boldsymbol{r}}_{mig_i}(T_j)$ is obtained as

$$\begin{aligned}\hat{\boldsymbol{r}}_{mig_i}(T_j) = &\boldsymbol{r}_{delay_i}(T_j) \\ &- min_{\mathbb{M}_i}\left\{\hat{\boldsymbol{r}}_{mt_{\mathbb{M}_i}}(T_j) + k\hat{\boldsymbol{r}}_{pc_{\mathbb{M}_i}}(T_j)\right\},\end{aligned} \quad (12)$$

where $\boldsymbol{r}_{delay_i}(T_j)$ is the QoS loss of s_i during T_j and $\mathbb{M}_i$ is the migration set of s_i ($\mathbb{M}_i \subset \Psi_i(T_j), \sum_{v_k \in \{\Psi_i(T_j) - \mathbb{M}_i\}} \mathfrak{r}_k(t_0+jT) \leq c_i$, where $\Psi_i(T_j)$ is the set of VMs located in s_i at the beginning of T_j).

Besides, $\boldsymbol{r}_{delay_i}(T_j)$ is calculated as

$$\begin{aligned}\boldsymbol{r}_{delay_i}(T_j) = &\sum_{v_k \in \Psi_i(T_j)} [\hat{\mathfrak{p}}_k(T_j) \\ &-\mathfrak{p}_k\left(t_0 + jT + \frac{1 - \mathfrak{pr}_k(t_0+jT)}{\frac{\mathfrak{pr}_k(t_0+jT)}{jT - t_{start_k}} \times \frac{a_i(t_0+jT)}{r_i(t_0+jT)} \times \frac{1}{th_{over}}}\right)\Bigg],\end{aligned}$$

where $r_i(t) = \sum_{v_k \in \Psi_i(T_j)} \mathfrak{r}_k(t)$.

We find the best migration set $\mathbb{M}_i$, which obeys

$$\min_{\mathbb{M}_i} \hat{\boldsymbol{r}}_{mt_{\mathbb{M}_i}}(T_j) + k\hat{\boldsymbol{r}}_{pc_{\mathbb{M}_i}}(T_j).$$

As the number of VM in a server is relatively small, we simply enumerate some subsets of $\Psi_i(T_j)$ to find such $\mathbb{M}_i$. Since the destination can be empty server, there exists two cases in computing $\hat{\boldsymbol{r}}_{mt_{\mathbb{M}_i}}(T_j) + k\hat{\boldsymbol{r}}_{pc_{\mathbb{M}_i}}(T_j)$.

Case i: All the VMs in $\mathbb{M}_i$ can be migrated to other non-empty servers without any over-utilized server addition. In this case, $\hat{\boldsymbol{r}}_{mt_{\mathbb{M}_i}}(T_j)$ and $\hat{\boldsymbol{r}}_{pc_{\mathbb{M}_i}}(T_j)$ can also be calculated as Eq. (10) and (11), where w_k is similarly obtained as

$$w_k = \max\left\{0, th_{over} - \frac{a_k(t_0+jT) + \frac{\sum_{v_l \in \mathbb{M}_i} \mathfrak{r}_l(t_0+jT)}{\#VM\ in\ \mathbb{M}_i}}{c_k}\right\}. \quad (13)$$

Case ii: Some VMs need to be migrated to some empty servers. $\mathbb{M}_i$ is separated into two parts ($\mathbb{M}_i = \{\mathbb{M}_{i1}, \mathbb{M}_{i2}\}$), where only VMs in $\mathbb{M}_{i1}$ can be migrated to non-empty servers. For $\mathbb{M}_{i1}$, $\hat{\boldsymbol{r}}_{mt_{\mathbb{M}_{i1}}}(T_j) + k\hat{\boldsymbol{r}}_{pc_{\mathbb{M}_{i1}}}(T_j)$ can be also calculated by Eqs. (10) and (11), where w_k is obtained in (13). For $\mathbb{M}_{i2}$, a set of empty servers $\mathbb{N}_i$ need to be found in advance ($\mathbb{N}_i \subset S$), which satisfies

$$\mathbb{N}_i = arg\min_{\mathbb{N}_i}\left[\hat{\boldsymbol{r}}_{mt_{\mathbb{M}_{i2}}}(T_j) + k\left(\hat{\boldsymbol{r}}_{pc_{\mathbb{M}_{i2}}}(T_j) + \sum_{s_k \in \mathbb{N}_i} p_{e_k}\right)\right]. \quad (14)$$

In this case, $\min_{\mathbb{M}_i}\left\{\hat{\boldsymbol{r}}_{mt_{\mathbb{M}_i}}(T_j) + k\hat{\boldsymbol{r}}_{pc_{\mathbb{M}_i}}(T_j)\right\}$ satisfies

$$\begin{aligned}\min_{\mathbb{M}_i}\left\{\hat{\boldsymbol{r}}_{mt_{\mathbb{M}_i}}(T_j) + k\hat{\boldsymbol{r}}_{pc_{\mathbb{M}_i}}(T_j)\right\} &= \hat{\boldsymbol{r}}_{mt_{\mathbb{M}_{i1}}}(T_j) + \hat{\boldsymbol{r}}_{mt_{\mathbb{M}_{i2}}}(T_j) \\ &+ k\left(\hat{\boldsymbol{r}}_{pc_{\mathbb{M}_{i1}}}(T_j) + \hat{\boldsymbol{r}}_{pc_{\mathbb{M}_{i2}}}(T_j) + \sum_{s_k \in \mathbb{N}_i} p_{e_k}\right).\end{aligned} \quad (15)$$

After calculating $\boldsymbol{r}_{delay_i}(T_j)$ for each server, we try to migrate the VMs of the servers from high estimated migration revenue to low estimated migration revenue until some servers can not be migrated or get negative revenue. The destination of each migrated v_i is obtained as

$$\begin{aligned}des_i = arg\min_{s_k \in \Lambda_i} &\frac{\|s_k - \mathfrak{l}_i(t_0+jT)\|}{3 \times 10^8 m/s} \\ &+ k(p_{m_k} - p_{e_k}) \times \mathfrak{r}_i(t_0+jT) + \iota,\end{aligned} \quad (16)$$

where Λ is the set of feasible destination servers and ι is a random variable.

IV. Experiment

A. Experiment Settings

Our algorithm is implemented in the CloudSim simulator [11]. We create 25 data centers and deploy them in a 5×5 grid in simulator. The distance between adjacent grid is $1km$. In each data center, there exists 5 heterogeneous servers of two types: HP ProLiant G4 and HP ProLiant G5.

We randomly choose 125 workload of nine days during 2011 of the CoMon project [14]. We test our algorithm in both differentiated and non-differentiated services. In non-differentiated services, we assume that $\mathfrak{p}_1(t_0+jT) = ... = \mathfrak{p}_N(t_0+jT) = 500 - j$, while we put tasks to five types ($type = \{1, 2, ..., 5\}$) randomly in differentiated services. The QoS revenue of each type is $\mathfrak{p}(t_0+jT) = 500 - \frac{1}{8} \times 2^{type} \times j$.

As for parameters, for each v_i, $\mathfrak{mi}_i = 2.16 \times 10^8$. Besides, we set the parameter $\mathcal{M} = 10^6, \mathcal{N} = 10^3$ of Ω, $th_\Gamma = 0.85, th_\mathcal{D} = 2$, and $th_{over} = 0.9$ [15]. For these three types of switches, we set $th_{ToR} = 10Gbps, th_{aggr} =$

Algorithm 1 Adaptive resource management

```
Input: Learning parameter $\alpha, \epsilon, \gamma, \iota$
Output: Action $\mathcal{A}$
 1: Initialize $\Omega$ to be empty list
 2: for each time slice $T_j$ do
 3:     Observe the current state $\mathcal{S}_j$
 4:     Calculate reward $\boldsymbol{r}_{j-1}$
 5:     Update corresponding Q-value $\mathcal{Q}$ by Eq. (7)
 6:     $similarStateFlag = false$
 7:     if $\exists\omega$ similar with $\mathcal{S}_j$ then
 8:         $similarStateFlag = true$
 9:     else if $|\Omega| < \mathcal{M}$ then
10:         Add this state to $\Omega$
11:         $similarStateFlag = true$
12:     else
13:         Replace the most similar state with $\mathcal{S}_j$
14:     end if
15:     Generate random number $\varphi$
16:     if $\varphi \geq \epsilon$ then
17:         $\mathcal{A}_j = randomActionSelection(\iota)$
18:         if $\exists$ action similar with $\mathcal{A}_j$ then
19:             Find the most similar action
20:         else if $|\omega| < \mathcal{N}$ then
21:             Add this action to $\omega$
22:         else
23:             Replace the most similar action with $\mathcal{A}_j$
24:         end if
25:     else
26:         Find action $\mathcal{A}_j$ with maximum revenue and sufficient
            transit network bandwidth
27:         if $\mathcal{A}_j = \emptyset$ then
28:             goto 17
29:         end if
30:     end if
31: end for
```

$20Gbps, th_{core} = 40Gbps$, which are traditional settings shown in Singh *et al.* [16].

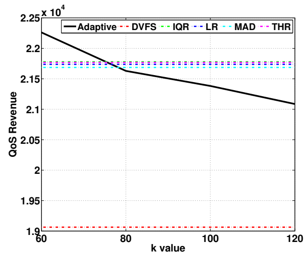
We compare our algorithm with dynamic voltage and frequency scaling (DVFS), static threshold (THR), median absolute deviation (MAD), interquartile range regression (IQR) and local regression (LR) algorithms proposed in Beloglazov *et al.* [15].

B. Experiment Results

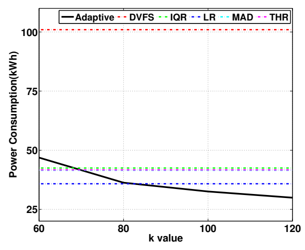
*1) **Performance with different** k:* The algorithm performance with different k of non-differentiated and differentiated services of sample workload on Mar. 3^{rd} is shown in Fig. 2 and 3, respectively. Besides, the total revenue of non-differentiated and differentiated services are demonstrated in 4 (a) and 4 (b), respectively. Compared with existing algorithms, our algorithm can adapt the change of k. In Fig. 2 (a) and 3 (a), when k is small enough, the QoS revenue of our algorithm is bigger than existing algorithms. Besides, in Fig. 2 (b) and 3 (b), when k is big enough, the power consumption of our algorithm is smaller than existing algorithms. In Fig. 4 (a) and 4 (b), our algorithm is more robust than the existing methods,

Algorithm 2 Random action selection

```
Input: Learning parameter $\iota$
Output: Action $\mathcal{A}$
 1: $migrateFailFlag = false$
 2: repeat
 3:     for each server $s_k$ do
 4:         if $r_k(t_0 + jT) \leq th_{over}$ || $!similarStateFlag$ then
 5:             Update action $\mathcal{A}_j$
 6:             Calculate $\hat{\boldsymbol{r}}_{mig_i}(T_j)$ by Eq. (9)
 7:         else
 8:             Calculate $\mathbb{M}_k$ and $\mathbb{N}_k$ by Eq. (14) and (15)
 9:             Calculate $\hat{\boldsymbol{r}}_{mig_i}(T_j)$ by Eq. (12)
10:         end if
11:     end for
12:     Find server $s_{(k)}$ with maximum $\hat{\boldsymbol{r}}_{mig_i}(T_j)$
13:     if $r_{(k)}(t_0 + jT) \leq th_{over}$ then
14:         Calculate $des_l$ ($v_l \in \Psi_k(T_j)$) by Eq. (16)
15:     else
16:         Calculate $des_{l1}$ and $des_{l2}$ ($v_{l1} \in \mathbb{M}_{k1}, v_{l2} \in \mathbb{M}_{k2}$)
            by Eq. (16)
17:     end if
18:     if $\exists v_r, des_r = \emptyset$ then
19:         $migrateFailFlag = true$
20:     end if
21: until $migrateFailFlag$
22: end
```

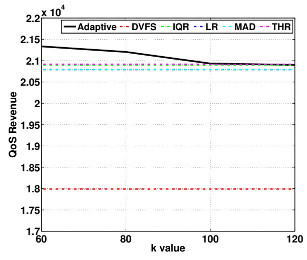


(a) QoS revenue

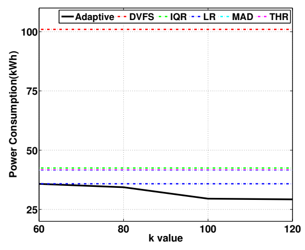


(b) Power consumption

Fig. 2. Performance with different k of non-differentiated services

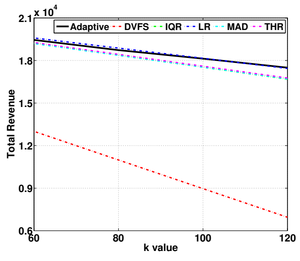


(a) QoS revenue

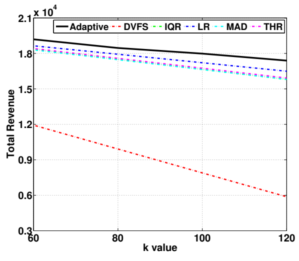


(b) Power consumption

Fig. 3. Performance with different k of differentiated services

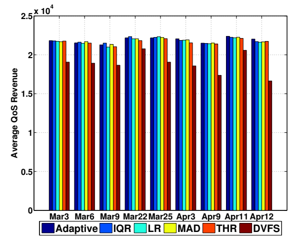


(a) Non-differentiated services

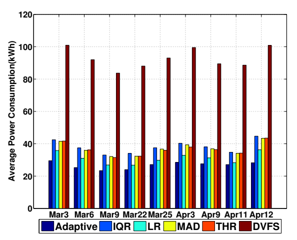


(b) Differentiated services

Fig. 4. Total revenue with different k

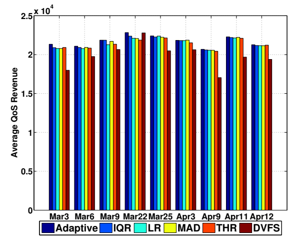


(a) Average QoS revenus

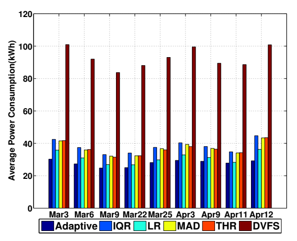


(b) Average power consumption

Fig. 5. Performance with different workload of non-differentiated services



(a) Average QoS revenus



(b) Average power consumption

Fig. 6. Performance with different workload of differentiated services

and it can achieve the better tradeoff between differentiated QoS revenue and power consumption.

*2) **Performance with different workload:*** Fig. 5 and 6 demonstrate the algorithm performance with different workload of non-differentiated and differentiated services of sample workload in Stone *et al.* [14]. Fig. 5 shows that with nearly equivalent average QoS revenue in non-differentiated services, our algorithm can reduce the average power consumption by 13.3%. Meanwhile, Fig. 6 further shows that our algorithm can cut off 9.6% power consumption in differentiated services.

V. Conclusion

In this paper, we study the adaptive resource management of differentiated services in geo-distributed data centers. The shortage of modeling QoS by SLA violation is analyzed, and we model it directly through differentiated revenue. A reinforcement learning based adaptive resource management algorithm is proposed and optimized for fast decision-making. Experiments show that our algorithm is more stable than other existing algorithms, and can get balance for differentiated services. In the future, we may consider the adaptive resource management of differentiated services in fog data centers, which is also a sort of geo-distributed data centers. Besides, we may do some research on deep reinforcement learning algorithms in finding such balance.

Acknowledgment

This work is supported by Chinese National Research Fund (NSFC) Key Project No. 61532013 and 61572262; National China 973 Project No. 2015CB352401; Shanghai Scientific Innovation Act of STCSM No.15JC1402400 and 985 Project of Shanghai Jiao Tong University with No. WF220103001.

References

[1] C. Gu, C. Liu, J. Zhang, H. Huang, and X. Jia, "Green scheduling for cloud data centers using renewable resources," in *INFOCOM WKSHPS*, 2015, pp. 354–359.

[2] M. Shatnawi and M. Hefeeda, "Real-time failure prediction in online services," in *INFOCOM*, 2015, pp. 1391–1399.

[3] L. Rao, X. Liu, L. Xie, and W. Liu, "Coordinated energy cost management of distributed internet data centers in smart grid," *IEEE Transactions on Smart Grid*, vol. 3, no. 1, pp. 50–58, 2012.

[4] L. Yu, T. Jiang, and Y. Cao, "Energy cost minimization for distributed internet data centers in smart microgrids considering power outages," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 1, pp. 120–130, 2015.

[5] H. Lin, X. Qi, S. Yang, and S. Midkiff, "Workload-driven vm consolidation in cloud data centers," in *IPDPS*, 2015, pp. 207–216.

[6] F. P. Tso, K. Oikonomou, E. Kavvadia, and D. P. Pezaros, "Scalable traffic-aware virtual machine management for cloud data centers," in *ICDCS*, 2014, pp. 238–247.

[7] Z. Han, H. Tan, G. Chen, R. Wang, Y. Chen, and F. C. Lau, "Dynamic virtual machine management via approximate markov decision process," in *INFOCOM*, 2016, pp. 1–9.

[8] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *PDP*, 2014, pp. 500–507.

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[10] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[11] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[12] C. Fuerst, S. Schmid, L. Suresh, and P. Costa, "Kraken: Online and elastic resource reservations for multi-tenant datacenters," in *INFOCOM*, 2016, pp. 1–9.

[13] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," *arXiv preprint arXiv:1006.0308*, 2010.

[14] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the royal statistical society. Series B (Methodological)*, pp. 111–147, 1974.

[15] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[16] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 183–197, 2015.