Hindawi
Scientific Programming
Volume 2020, Article ID 1704258, 12 pages
https://doi.org/10.1155/2020/1704258

Research Article

An Adaptive Data Placement Architecture in Multicloud Environments

Pengwei Wang,[1] **Caihui Zhao**,[1] **Yi Wei**,[1] **Dong Wang**,[2] **and Zhaohui Zhang**[1]

[1]*School of Computer Science and Technology, Donghua University, Shanghai, China*
[2]*School of Computer Science & Information Engineering, Shanghai Institute of Technology, Shanghai, China*

Correspondence should be addressed to Pengwei Wang; wangpengwei@dhu.edu.cn and Dong Wang; dongwang@sit.edu.cn

Received 23 October 2019; Accepted 24 February 2020; Published 10 June 2020

Guest Editor: Aibo Song

Cloud service providers (CSPs) can offer infinite storage space with cheaper maintenance cost compared to the traditional storage mode. Users tend to store their data in geographical and diverse CSPs so as to avoid vendor lock-in. Static data placement has been widely studied in recent works. However, the data access pattern is often time-varying and users may pay more cost if static placement is adopted during the data lifetime. Therefore, it is a pending problem and challenge of how to dynamically store users' data under time-varying data access pattern. To this end, we propose ADPA, an adaptive data placement architecture that can adjust the data placement scheme based on the time-varying data access pattern and subject for minimizing the total cost and maximizing the data availability. The proposed architecture includes two main components: data retrieval frequency prediction module based on LSTM and data placement optimization module based on Q-learning. The performance of ADPA is evaluated through several experimental scenarios using NASA-HTTP workload and cloud providers information.

1. Introduction

With the development of cloud computing, more and more companies adapt the cloud to store their data for low maintenance costs and reliable SLAs (Service Level Agreements) comparing to the traditional data storage mode. Many mainstream cloud service providers (CSPs) offer a variety of data storage services to satisfy different users' demands. The pricing of services for the same functionality is diverse among cloud service providers (CSPs). And the pricing policy of the same CSP is various in different regions. The cost of data migration among data centers of the same CSP is cheaper than that in different CSPs.

In addition to the constraints of high migration costs, a single cloud faces the risk of vendor lock-in; i.e., major risks include the price of cloud services and the interruption of SLA. These situations may result in making users pay expensive migration costs. In our previous work, we have proposed an ant colony algorithm-based approach for cost-effective data hosting with high availability in multicloud environments [1]. In order to avoid vendor lock-in, we can divide the original data and store the data chunks into multiple CSPs. Furthermore, there is a more comprehensive solution to this problem in [2]. We aim at providing a cost-effective and high-available data placement in multicloud environments based on users' demands.

In the previous work [1–3], the workload of data object affects the choice of data placement scheme. The data workload is related to the data retrieving frequency (DAF) (i.e., Get access rate) during a fixed time period. In the life-cycle of data placement, the workload is time-varying. The data object with high data is read-intensive and in hot-spot status, which is more likely to be stored in CSPs with lower out-bandwidth costs additionally. Conversely, the data object with low DAF is storage-intensive and in cold-spot status, which is more likely to be stored in CSPs with lower storage costs [4]. If a user adapts a data placement scheme with a relatively low-accessible frequency for the entire life-cycle of data storage, it may generate more out-bandwidth costs when DAF increases. In another case, if a user employs

the strategies that are more suitable for hot-spot status in the entire life-cycle of data storage, it may produce expensive storage costs when DAF reduces.

In order to reduce the overall cost and enhance availability during the data object lifetime, it is essential to develop a mechanism to dynamically adjust the data placement scheme based on data object workload. Due to the uncertainty of future data workload, the overall cost cannot get the best result. Therefore, predicting the future workload is a key part of the dynamic data placement mechanism. When the future workload is obtained, how to design a dynamic placement scheme based on future access frequency becomes another significant component.

ADPA, which can dynamically host the data object with cost-effective and high availability based on time-varying data workload, is an adaptive data placement architecture proposed in our study. The key contributions of our study are as follows.

Firstly, we propose a LSTM [5] algorithm based on workload prediction. It can use historical workload data to predict future workload.

Secondly, a dynamic data placement algorithm based on reinforcement learning is proposed. It can migrate data according to the change of workload to ensure cost optimization and availability.

Finally, we evaluate ADPA through several experiment scenarios. The results show that ADPA algorithm not only is superior to SOA algorithm, but also can save more time than ACO and DP algorithm to obtain the optimal data placement.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents the motivation of the adaptive data placement. Section 4 describes the architecture of adaptive data placement. Section 5 introduces the proposed algorithm. The performance of our proposed algorithm is shown via extensive experiments by using real-world cloud information in Section 6. Finally, Section 7 summarizes this paper.

2. Related Work

There are many studies for data placement optimization in cloud computing. We can divide these studies into two categories based on whether the data storage scheme can be adjusted according to workload: static data placement and dynamic data placement. In static data placement, data placement scheme is determined in advance and does not adapt to variable data object workload. However, in dynamic data placement, data placement scheme can be obtained according to the change of DAF. In the following, we review and discuss them, respectively.

2.1. Static Data Placement. In [6], the authors present the fact that availability of service and data lock-in are two obstacles for cloud computing. In cloud storage, replication and erasure coding are widely used in increasing data availability [7]. To avoid data lock-in, multicloud plays an important role in data storage.

Mansouri et al. [8] propose an approach to replicate data objects in multiple data centers to enhance data availability and avoid data lock-in. In [9], the authors adapt erasure coding to ensure data security and minimize the total cost under many unreasonable assumptions. Hadji [10] focuses on minimizing data storage and presents two efficient algorithms to solve it. But the out-bandwidth cost is ignored. In these studies, each data item is at a fixed number in a static way; Qu et al. [11] present a resilient, fault-tolerance, and high-efficient global replication algorithm (RFH) to determine each data items' replication, migration, and suicide.

The above studies only optimize to minimize the monetary cost or enhance data availability under other QoS metrics. The trade-off between cost and availability is not considered. Wang et al. [1] propose an approach based on ant colony algorithm that minimizes total cost and enhances data availability through erasure coding. It is still a single objective optimization problem. In [12], Liu et al. use multiobjective particle swarm optimization algorithm to minimize storage space cost, data migration cost, and communication cost as well as enhancing the storage reliability. However, they cannot determine how to choose an optimal solution in the Pareto front.

All the above works only consider the static data placement and ignore the uncertainty of data object workload. Once the workload of data object is changed, the data placement scheme needs to be calculated again.

2.2. Dynamic Data Placement. The dynamic data placement in cloud storage becomes a hot direction in research. Zhang et al. [3] propose a data hosting scheme which contains a transition of storage modes based on changes of data access frequency. However, they only put forward a transfer condition without considering the global data placement sequence optimization problem. In [13], Su et al. present a systematic model, in order to formulate data placement optimization for complex requirements in multicloud environments. They also only discuss the data migration without making the final decision for developers.

Due to the unknown future data access pattern, the above studies do not propose the automatic data migration. Mansouri et al. [4] propose two online algorithms that make a trade-off residential and migration cost. Due to the absence of the future data object workload, the authors use the deterministic online algorithm to solve the problem.

To achieve a dynamic data replication strategy based on real data access pattern, Gill and Singh [14] propose a dynamic cost-aware rereplication and rebalancing strategy. This method firstly determines which file and when to replicate based on files popularity. The more the popular it is, the more possible the file is replicated. Then, it calculates the replica number to meet the availability requirements. Finally, a replica placement mechanism is proposed. Lyapunov optimization is an online control method to solve the problem which closes the optimal solution. In [15], Qiu et al. adapt Lyapunov for dynamic content distribution service

deployment without the need for any future request arrival information. In other works of dynamically optimized storage, T. Sreenath Reddy and G. MURALI [16] use GA-based totally records dissemination alternate technique and dynamic request redirection to minimize the cost of grouped gets. The algorithm IMPSO used in [17] also inspires our work.

Except for the above methods, a method based on the historical workload pattern is also effective to solve the dynamic data placement. In [18], Papaioannou et al. propose a cloud storage brokerage solution that can periodically recompute the best provider set using data access statics of the last sampling periods. It can adjust data placement for dynamically changing data access pattern. In recent years, reinforcement learning (RL) has received extensive attention. RL is a powerful approach for a long time decision-making under uncertainties [19]. As far as we know, there are no studies on using RL to optimize the dynamic data placement.

In this paper, we propose an adaptive data placement method that mainly contains data retrieval frequency prediction and data placement optimization. In data retrieval frequency prediction module, we adapt LSTM to predict the future workload of data object. In data placement optimization module, an approach based on Q-learning is used to get a sequential data placement solution according to the prediction data object workload.

3. Motivation

3.1. Dynamic Data Object Workload. The DAF of data object is time-varying. We collect the trace of NASA-HTTP that depicts all HTTP requests to the NASA Kennedy Space Center WWW server in Florida [24]. We calculate the DAF in a cycle of 10 minutes. As shown in Figure 1, it depicts the DAF's change from 01/Jul/1995 : 0:0 : 0 to 07/Jul/1995 : 23 : 59 : 59. The difference between the maximum DAF value and the minimum DAF value is close to 2. DAF was relatively low at night but increased during the day. The data placement scheme needs to be dynamically adjusted according to the changing DAF, which will be discussed in Section 3.3.

3.2. Heterogeneous Cloud Market. Now there are many CSPs providing storage services, and we collect four most popular CSPs' pricing strategies: Amazon S3 [20], Microsoft Azure Cloud Storage [21], Alibaba Cloud Object Storage [22], and Google Cloud Storage [23] as shown in Table 1. From the point of users' cost, the selection of CSP is a research direction in cloud computing. In our previous work, we proposed some approaches for optimizing selection of cloud instance types [25–27]. Actually, the price of the same functional storage service provided by the same CSP in different regions is different. And the price of the same functional storage service across CSPs is also different. For example, Amazon S3 in New York, USA, has lower storage price than that in Tokyo. For instance, Amazon S3 in New York has lower storage cost than Microsoft Azure Cloud Storage in New York.

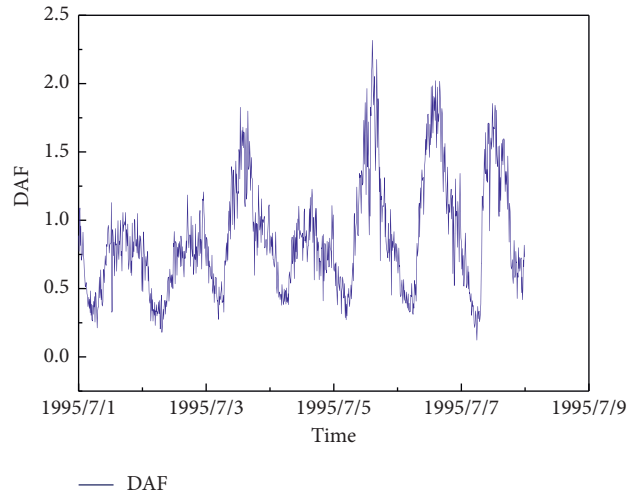

Figure 1: NASA-HTTP requests in ten-minute cycles from 01/Jul/1995 to 07/Jul/1995.

3.3. Discussions. The dynamic data object workload and heterogeneous cloud market inspired us to propose an adaptive data placement scheme. Then, we discuss this in detail through the NASA-HTTP example.

When the access frequency is low, the data is suitable for CSPs with low storage cost. But the bandwidth cost of these cloud service providers may be relatively high. We assume that Amazon S3 in New York, Tokyo, and London is chosen as the data placement scheme because of the lower storage cost. The out-bandwidth price of Amazon S3 in New York, Tokyo, and London is very expensive. As time goes by, DAF gradually becomes larger and the output bandwidth price will be very expensive. If users migrate their data to CSPs with lower out-bandwidth cost, they can save more money even if they need to pay for additional migration cost.

In this study, we propose an adaptive data placement scheme which dynamically adjusts the DAF-based data storage scheme to minimize the total cost.

4. An Adaptive Data Placement Architecture

In this section, we present an adaptive data placement architecture with high cost-effective multicloud availability. Then, we formulate the problem in the architecture.

4.1. Architecture Overview. Figure 2 depicts the architecture of dynamic data placement. There are four components: *Cloud Storage Information Collection*, *Optimization Module*, *Workload Statistics Module*, and *Prediction Module*.

Cloud Storage Information Collection is used to collect the information of CSPs including storage price, out-bandwidth price, and operation cost. Apart from this, it receives DAF prediction to adjust data placement scheme.

Optimization Module is used to optimize the data placement scheme according to users' demands which

Table 1: Pricing of storage (in $/GB/month), Out-bandwidth (in $/GB/month), and get requests (in $/10 K/month) of each CSP.

CSP	Amazon S3 [20]			Microsoft Azure Cloud Storage [21]			Alibaba Cloud Object Storage [22]			Google Cloud Storage [23]		
	New York	Tokyo	London	New York	Dublin	Hong Kong	Beijing	San Francisco	Sydney	Atlanta	St. Ghislain	Changhua County
Storage price	0.0125	0.019	0.0131	0.0208	0.022	0.024	0.0226	0.02	0.0209	0.026	0.026	0.026
Out-bandwidth price	0.05	0.12	0.05	0.02	0.02	0.9	0.117	0.076	0.13	0.02	0.02	0.2
Get request price	0.004	0.0037	0.0042	0.004	0.004	0.004	0.001	0.001	0.002	0.004	0.004	0.004

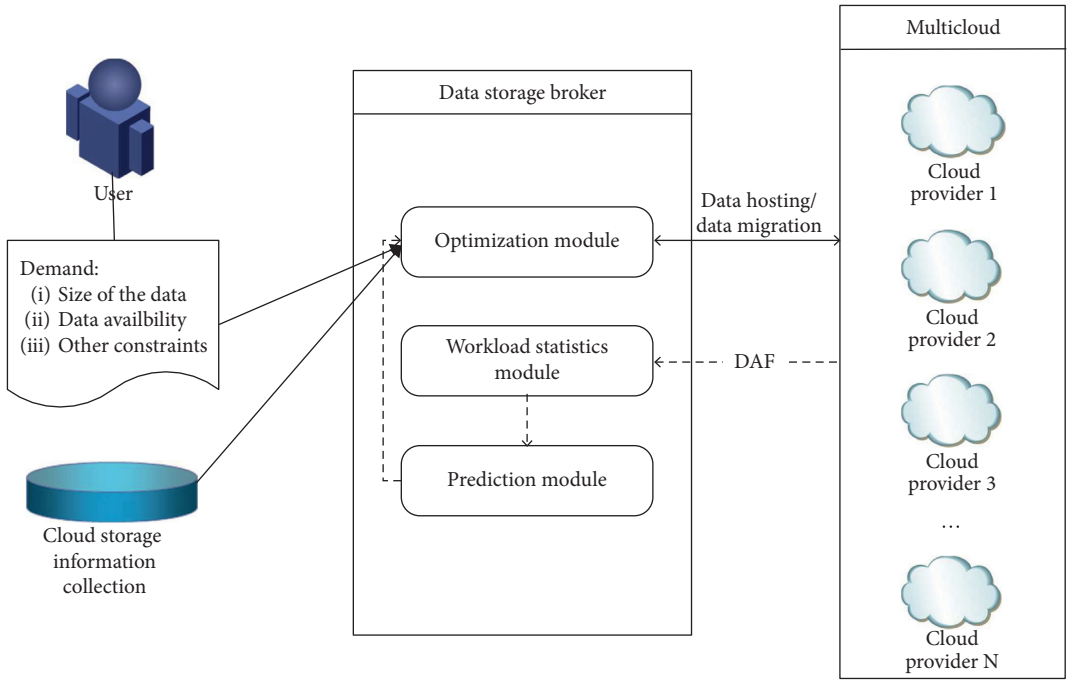

Figure 2: The architecture of ADPA.

contain the size of data object, required availability, initial DAF, and so on.

Workload Statistics and Prediction Module are responsible for collecting the historical data of workload and were based on these historical data to predict the DAF for the next period of time.

4.2. Problem Definition. To describe a dynamic data placement architecture in detail, we introduce the following definitions. The symbols used in this paper are listed in Table 2.

Definition 1 (data center specification). Assume that there are N data centers $\mathrm{DC} = \{d_1, d_2, \ldots, d_N\}$. Each data center d has tuple: P_d^s, P_d^b, P_d^o, a_d, where a_d defines the probability of data center d being available.

Definition 2 (data object specification). Assume that there is a data object with the size S and the DAF $r(t)$ where $t \in [1, T]$, the required availability is A_{req}, and the required data retrieval latency is L_{req}.

In this work, we also adapt erasure coding to avoid vendor lock-in, enhance data availability, and reduce storage and out-bandwidth costs compared to replication. It is worth noting that the following definitions for availability and cost are similar to those in [1, 3, 13], which is a universal way to define them for data hosting in erasure coding mode. We give the definition of erasure coding at first.

Definition 3 (erasure coding). An (m, n)-erasure coding denotes the data object which is divided into m chunks and encodes m chunks into n chunks. Users can retrieve their data through any m chunks.

Definition 4 (data availability). Since users can tolerate $0 \sim (n - m)$ shut-down of data centers at the same time, data availability is the sum probability of all cases that k DCs are simultaneously available, $k \in [m, n]$. Assume that $D(t)$ denotes the data centers chosen for data storage at time slot

Table 2: Symbol table.

Symbols	Descriptions
DC	List of data centers
N	Total number of data centers
P_d^s	Storage price of data center d
P_d^b	Out-bandwidth price of data center d
P_d^o	Operation price of data center d
L_d	Location of data center d, which contains latitude and longitude
T	Number of time slots
$r(t)$	The data access frequency at time slot t
A_{req}	The required availability
L_{req}	The required data retrieval latency
(m, n)	The parameters of erasure coding
S	Size of the data object
$D(t)$	Data placement solution at time slot t
$A(t)$	The availability of the solution of time slot t
$P_{\text{stor}}(t)$	Storage cost of a solution at time slot t
$P_{\text{net}}(t)$	Network cost of a solution at time slot t
$P_{\text{op}}(t)$	Operation cost of a solution at time slot t
$C_B(t)$	The sum of storage cost, network cost, and operation cost at time slot t
$C_M(t)$	The migration cost
$D_r(t)$	Data centers for data retrieval at time slot t
$l(t)$	Data retrieval latency at time slot t
D	The candidate data placement solutions
D^*	The optimal data placement during $[1, T]$

t, where $D(t) = \{d_1, d_2, \ldots, d_n\}$. We use $\Theta = \binom{|D(t)|}{k}$ to indicate the number of cases that k data centers are available. $D_j^\Theta(t)$ denotes the jth data centers collection in θ cases at time slot t. The data availability at time slot t can be defined as follows:

$$A(t) = \sum_{k=m}^{n} \sum_{j=1}^{\Theta} \left[\prod_{d \in D_j^\Theta(t)} a_d \prod_{d \in D(t) \backslash D_j^\Theta(t)} (1 - a_d) \right], \tag{1}$$

where $D(t) \backslash D_j^\theta(t)$ is the data centers that are not available.

Definition 5 (storage cost). the storage cost is the sum of storage cost of n data centers. It can be calculated as

$$P_{\text{stor}}(t) = \sum_{d \in D(t)} \frac{S}{m} P_d^s, \tag{2}$$

where S/m denotes the size of data stored in each data center.

Definition 6 (network cost). Users can retrieve their data through any m data chunks. We choose m data centers with the lowest out-bandwidth price to retrieve data. Since the DAF is time-varying, network cost is related to $r(t)$ of each time slot. It can be calculated as follows:

$$P_{\text{net}}(t) = \min_{j \in [1,\Theta]} \left(\sum_{d \in D_j^\Theta(t)} \frac{S}{m} r(t) P_{bi} \right). \tag{3}$$

Definition 7 (operation cost). The operation cost can be defined as follows:

$$P_{\text{op}}(t) = \min_{j \in [1,\Theta]} \left(\sum_{d \in D_j^\Theta(t)} r(t) P_{oi} \right). \tag{4}$$

It is worth noting that the value of j in (3) and (4) is equal.

Definition 8 (base cost). The base cost $C_b(t)$ is the sum of storage cost, network cost, and operation cost at time slot t. It can be defined as

$$C_B(t) = P_{\text{stor}}(t) + P_{\text{net}}(t) + P_{\text{op}}(t). \tag{5}$$

Definition 9 (migration cost). Different DAFs may correspond to different optimal data placement solutions. It generates expensive cost if users adapt the previous solution. Therefore, it can save more cost according to the DAF dynamically adjustment data placement. But the data placement adjustment means data migration which also requires cost. Data migration is not the moment when DAF changes but satisfies the conditions of the cost saved by the new solution (compared to the old one) which can cover the migration cost. We use $D' = D(t-1) \cap D(t)$ to indicate the intersection of data placement solutions of time slots t and $(t-1)$. The data centers that need data migration are $D(t-1)/D'$. The migration cost can be defined as follows:

$$C_M(t) = \sum_{d \in D(t-1) \backslash D(t)} \left\{ \frac{S}{m} P_d^b + P_d^o \right\}, \tag{6}$$

where the condition is as follows:

$$C_B(t-1) - C_B(t) \geq C_M(t). \tag{7}$$

Definition 10. (data retrieval latency). The data centers for data retrieval $D_r(t)$ at time slot t are m data centers with the lowest out-bandwidth cost. The data retrieval latency is the maximum latency of $D_r(t)$. Since the data retrieval latency is determined by network latency, we use round-trip time to indicate data retrieval latency [4, 28, 29]. We use the calculation method in [30], as follows:

$$l(t) = \max_{d \in D_r(t)} \{5 + 0.02\text{Distance}(d)\}, \tag{8}$$

where $\text{Distance}(d)$ is the distance between users and data centers.

4.3. Optimization Problem. In ADPD, it aims to obtain the data placement solution $D(t)$ in each time slot so that the total cost during $t \in [1, T]$ is minimized. The optimization problem is defined as follows:

$$\min_{D(t)} \sum_{t \in [1,T]} (C_B(t) + C_M(t)), \tag{9}$$

subject to

(1) $|D(t)| = n$, $\forall t \in [1, T]$

(2) $A(t) \geq A_{\text{req}}$, $\forall t \in [1, T]$

(3) $l(t) \leq L_{\text{req}}$, $\forall t \in [1, T]$

Constraint (1) indicates that only n data chunks in each time slot. Constraints (2) and (3) ensure that the data placement solutions satisfy the user required availability and data retrieval latency in each time slot. Obviously, this optimization problem is NP-hard.

5. Solution

In this section, we describe the implementation of ADPD. Firstly, we adapt LSTM for data object DAF prediction. In order to make decision-making effectively, the method to solve the problem is mainly based on Q-learning that is an off-policy temporal difference (TD) control algorithm [31, 32].

5.1. DAF Prediction. Recently, there are many machine learning approaches including Support Vector Machines (SVMs), Linear Regression (LR), Random Forest (RF), and k-Nearest Neighbors (kNNs) adopted for prediction. In this study, we use LSTM to predict the future DAF. We can also use other more accurate algorithms to replace LSTM Algorithm 1.

Since our previous work is to predict the price of cloud spot instance whose problem is also for time series data prediction [33]. In this study, we also use sliding window to split the historical DAF data. For DAF prediction, its goal is to find a function f that used historical DAF to predict the future DAF.

5.2. Data Placement Optimization. In this section, the data placement optimization aims to obtain the optimal solution $D(t)$ according the future DAF. Assuming that $D = \{D_1, D_2, \ldots, D_{|D|}\}$ denotes the candidate data placement solutions at each time slot, where $|D| = \binom{|DC|}{n}$. Let D^* indicate the optimal data placement solution from $t = 1$ to $t = T$, where $|D^*| = T$. We describe the process of data placement during $[1, T]$ via Figure 3. For each $D_i \in D$, it has Markov property. The reason is that the solution choice for the next time slot is only related to the currently selected plan and is independent of the previous choices. It can be expressed as $P[D(t+1) \mid D(t)] = P[D(t+1) \mid D(1), D(2), \ldots, D(t)]$.

In data placement process, the cost directly determines the choice of the plan. And the data placement optimization is a sequential decision-making problem in our paper. So we can attribute it to Markov decision process (MDP) and solve it through reinforcement learning (RL). The MDP of data placement process can be defined as follows [34].

Definition 11 (MDP of data placement process). We can transfer data placement process into MDP, which is a 5-tuple one:

(1) A finite set of states is denoted by D

(2) A finite set of actions A are the choice of solution of next time slot

(3) $P^a_{DD'} = P[D(t+1) = D' \mid D(t) = D, A(t) = a]$ is the probability that data placement D at time slot t when taking choice a results in solution D' at time t+1

(4) A reward function $R(D, a, D')$ is the negative value of solution transitions total costs from D to D' when taking action a between two time slots

(5) The discount factor $\lambda \in [0, 1]$, which is the attention of future rewards

The goal of RL is to find the optimal policy based on MDP. The basic architecture of RL is shown in Figure 4 [32]. The agent not only interacts with the environment through making action according to the current state, but also gets an immediate reward and the state of observation [32]. RL aims to find an optimal policy to maximize the total rewards that are obtained from every step. In this optimization problem, the reward function is defined as follows:

$$R(t) = -C_B(t) - C_M(t). \tag{10}$$

So, the RL objective of this problem can be presented as

$$J_{D^*} = \max_D E_D \left[\sum_{t=1}^{T} \lambda^t r(t) \right], \tag{11}$$

where $\lambda = 1$ and D^* is the optimal data placement solution during time slot $[1, T]$.

For the above problem, Q-learning is widely used tabular RL algorithm [19, 31] which is an off-policy TD control. It is defined by [31]

$$Q(S, A) \longleftarrow Q(S, A) + \alpha[R + \lambda \max_a Q(S', a) - Q(S, A)], \tag{12}$$

where $R = R(t)$ that is defined in (10) and α is the learning rate.

As stated previously, we transfer the time series data placement optimization into an MDP and adapt the approach based on Q-learning to solve it. For that purpose, we proposed an MDP that comes from data placement optimization being the most important approach. Algorithm 2 represents the transfer methodology. In order to satisfy the constraints (1)–(3) in (9), we filter all candidate data placement solutions according to their availability and latency (Line 5). In the whole life-cycle of data storing, the DAF is time-varying, which results in the fluctuation of cost. Then, we calculate the state matrix that contains basic cost for all satisfied solutions (i.e., $D = \{D_1, D_2, \ldots, D_{|D|}\}$) at each time slot (Lines 6–9). Finally, the migration cost between two solutions at time slots t and $t-1$ needs to be calculated as the transfer matrix which is a $|D| \times |D|$ matrix (Lines 12–16). Since the migration cost between the same two solutions is free, the values of the diagonal on the matrix are 0.

Compared with general MDP, the reward function contains not only the value between states, but also the value of states which is defined in (10). Based on the above MDP, we present an approach based on Q-learning. The core of the method is calculation and update of Q tabular which is the basis for the agent to choose the next time slot's action. Since this paper solves a time-based sequential decision problem, the Q tabular also has a time dimension. The strategy of Q tabular is based on (12). The value function of next state is adapted to update the current state, which is called bootstrapping [32]. The approach based on Q-learning is an off-policy method; that is, the action strategy and the target strategy are different. In Q-learning, the action chosen strategy is ϵ-greedy and the target is greedy. Algorithm 3 represents the data placement methodology based on Q-learning.

6. Evaluation

In this section, we firstly describe the dataset used in experiments. Then, we present a basic algorithm to verify the necessity of future DAF prediction. Since ant colony optimization algorithm (ACO) and genetic algorithm (GA) are widely used to solve optimization problem [1], we compare ADPA with them through different experimental scenarios.

6.1. Settings. In this work, the setup for DAF, data centers' specification, and experiment parameters are as follows.

6.1.1. DAF. We use the real workload trace of NASA-HTTP from [24] which contains two months' worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida from 01/Jul/1995 to 31/Aug/1995. Each piece of data in NASA-HTTP represents a request, including host, timestamp, request, HTTP reply code, and bytes in the reply. In our paper, in order to obtain data access frequency, we count the number of *Get* operations in a specific time period, which is 600 seconds. The trace is separated as a train and test set for the LSTM at a ratio of 7: 3, respectively.

6.1.2. Data Center Specification. The real-world information of data center, including storage price, out-bandwidth price, *Get* operation price, and latitude and longitude, is collected from CSPs official website [20, 22, 23, 35]. We use 18 data centers from different CSPs, and each CSP's data center is located in a different city. We also simulate the availability values of each data center in the [95.0%, 99.9%] interval.

6.1.3. Experiment Parameters. In the experiments, we choose $(3, 5)$-erasure coding as the data splitting method. The size of sliding window for LSTM is 12. The size of data object is initially 200 GB. The required availability and data retrieval latency are initially 99.9% and 1000 ms, respectively.

6.2. Performance of ADPA. In the time-varying DAF data placement, we cannot obtain the global data placement solutions because of the absence of future DAF. In our method, we firstly predict future DAF for the global optimal data placement solutions during the life-cycle of data storage. In order to verify the performance of ADPA, we propose a step-by-step optimization algorithm called SOA as follows.

6.2.1. SOA Algorithm. This is shown in Algorithm 4. This method minimizes the total cost through Greedy. At the beginning of time slots, the method finds the data placement solution with the lowest basic cost in all candidate solutions. Then, the data objects are allowed to the solution with the lowest total cost at the current time slot (Line 3). In fact, the optimal solution of SOA is locally optimal.

Based on the DAF prediction, we can obtain the global optimal data placement solutions, but the solutions of Greedy are locally optimal. In order to verify this situation, we study the total costs of ADPA and SOA by varying data size and data retrieval latency, respectively, as shown in Table 3 and 4. Obviously, the results of ADPA can save \$13.566, \$26.693, \$38.78, \$52.433, and \$65.437 comparing to Greedy with data size from 100 GB to 500 GB, respectively. Due to the absence of future DAF, SOA only chooses the best solution at current time slot and ignores the importance of future total cost. Conversely, ADPA, based on Q-learning, chooses the current solution which is based not only on current costs, but also on future reward.

As shown in Table 4, we study the results of two methods with the data retrieval latency constraint from 200 ms to 500 ms, the number of time slots is 12 (i.e., 2 hours), and data size is 200 GB. As latency constraint increases from 200 ms to 500 ms, the cost saving of ADPA is 18.6%. When data retrieval latency is loose, comparing with strict latency, the method explores the solution with lower cost in a larger

Input: The required length max_step of the series to be forecast, a trained LSTM model L_{trained}.
Output: Predicted future DAF;
(1) cnt = 0;
(2) Feed the data in the last window into L_{trained} and get a prediction $L(\text{SW}_{\text{cnt}})$;
(3) **while** cnt < max_step **do**
(4) Slide the window forward by one step and put the newly predicted value $L(\text{SW}_{\text{cnt}})$ at the end of the slide window;
(5) Feed the data in the new window into L_{trained} and get another prediction $L(\text{SW}_{\text{cnt}})$;
(6) cnt = cnt + 1;
(7) **end while**

Algorithm 1: Algorithm LSTM: LSTM network with sliding window.

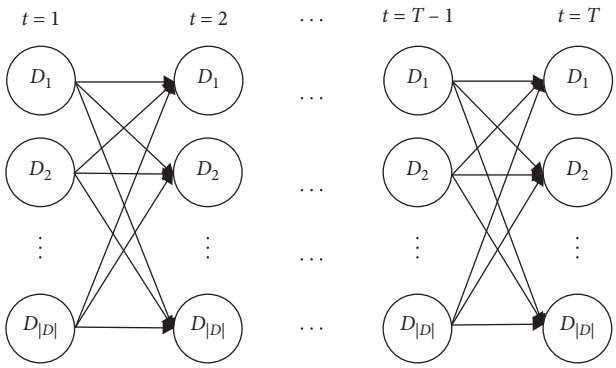

Figure 3: The data placement during the period $[1, T]$.

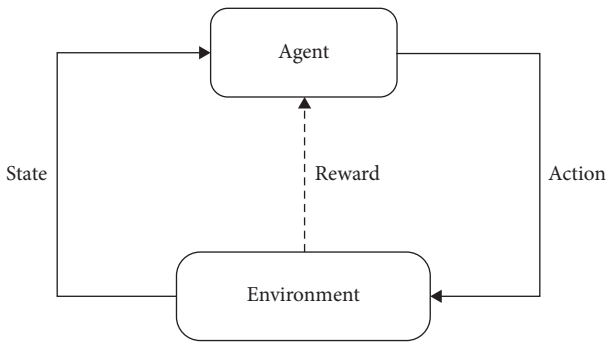

Figure 4: The basic architecture of RL.

range. The algorithm explores solutions in the area near users to satisfy the strict latency constraint.

In actual life, there are different levels of data access and these situations have different usage scenarios. In order to verify the universality of the ADPA algorithm, we extend the time range of NASA-HTTP request data to one year (24/Oct/1994–11/Oct/1995) and perform DAF statistics based on one-day cycles. Due to the wide time range of the second data set, it has a lower DAF than our first data set. The experimental environment is the same; we use the second dataset to conduct a comparison experiment as shown in Tables 5 and 6. Similarly, it is clear that ADPA can help users save more money.

6.3. Performance Comparison with Other Methods. We compare our method with ACO and DP through the following experimental scenarios. Assume that ACO and GA can obtain the DAF prediction which is predicted by ADPA.

6.3.1. Cost Saving with Varying Data Size. In this experimental scenario, we evaluate the cost performance of ADPA when data size is increased from 100 GB to 500 GB. From the experimental results, we can know that the total cost of three algorithms increases when the data size increases, and the costs of the three algorithms obtained under different data sizes are the same. Since the DP algorithm can find the optimal solution, it can prove the correctness of ADPA in solving the solution. To demonstrate the advantages of ADPA, the time required to find the optimal solution for the three algorithms is shown in Figure 5. It can be seen that the time required by ACO is greater than that of ADPA and DP algorithm. The algorithm ADPA is higher than DP when the data size is 200 GB, and the others are better than DP.

6.3.2. Cost Saving with Varying Time Slot Count. In this scenario, we explore the impact of time slot count (T) on total costs when T varies from 10 to 18. From these experimental results, we can know that the greater the T value, the longer the optimization time and the costs of the three algorithms obtained under different time slot count (T) are the same. Because of the same reason as above, this proves that ADPA has the ability of solving the solution. In order to reflect the advantages of ADPA, the time required to find the optimal solution for the three algorithms is shown in Figure 6. It can be seen that, for ADPA and DP, except for $T = 10$, the running time of the algorithm proposed in this paper is only 50% of the DP algorithm; that is, ADPA can solve the optimal solution in the least time.

6.3.3. Cost Saving with Varying Candidate Data Centers. We investigate the impact of the number of candidate data centers (N) on total cost. Figure 7 gives the results when N increases from 12 to 15. The change in candidate data centers count is adding a new data center in the previous set. For example, when $N = 10$, the index of data centers is $DC = \{3, 9, 16, 15, 14, 6, 7, 12, 8, 4, 0, 10\}$ and we add a new data center with index of 13 to this set when $N = 11$. When N

Input: Data center specification, DC, DAF, $r(t)$, the required availability, A_{req}, the required data retrieval latency, L_{req}.
Output: The state matrix, SM, the transfer matrix, TM.
(1) $D \longleftarrow$ Calculate all n-combinations of data centers DC.
(2) **for all** $c \in D$ **do**
(3) Calculate availability of c $\text{ava}(c)$ through (1);
(4) Calculate latency of c $\text{latency}(c)$ through (8);
(5) **if** $\text{ava}(c) \geq A_{\text{req}}$ and $\text{latency}(c) \leq L_{\text{req}}$ **then**
(6) **for** $t = 1$ to T **do**
(7) Calculate the base cost of c $C_B^c(t)$ through (5);
(8) $\text{SM}[c][t] = C_B^c(t) + C_M^c(t)$;
(9) **end for**
(10) **end if**
(11) **end for**
(12) **for all** $c1 \in D$ **do**
(13) **for all** $c2 \in D$ **do**
(14) $\text{TM}[c1][c2] \longleftarrow$ Calculate the migration cost between $c1$ and $c2$ through (6);
(15) **end for**
(16) **end for**
(17) **return** SM, TM;

Algorithm 2: Algorithm TDM: transform data placement into an MDP.

Input: The state matrix, SM, the transfer matrix, TM.
Output: The optimal data placement solution during $t \in [1, T]$, D^*
(1) Initialize parameters of algorithm including learning rate α, discount
(2) Initialize the tabular Q with zero;
(3) $s \longleftarrow$ Initialize the start state;
(4) Initialize data placement sequences Seq;
(5) **for** $e = 1$ to epochs **do**
(6) **for** $t = 1$ to T **do**
(7) $f\text{Set} \longleftarrow$ Choose feasible data placement solutions through equation
(8) Choose a data placement solution $D(t)$ from $f\text{Set}$ through ϵ-greedy function;
(9) Append $D(t)$ in Seq$[e]$;
(10) Obtain the next state s_{nxt}, reward $r(t)$, and the next state's
(11) $Q(t, s, D(t)) = Q(t, s, D(t) + \alpha * (-TM[s, D(t)] - SM[D(t), t] + \gamma * \max(Q[t + 1, s_{\text{nxt}}, A_{\text{nxt}}])) - Q[t, s, D(t)])$;
(12) $s = s_{\text{nxt}}$;
(13) Take ε decay;
(14) **end for**
(15) **end for**
(16) $D^* \longleftarrow$ Find the sequence with lowest cost in Seq;
(17) **return** D^*;

Algorithm 3: Algorithm DPQ: data placement algorithm based on Q-learning.

Input: The state matrix, SM, the transfer matrix, TM.
Output: The optimal data placement solution during $t \in [1, T]$,
(1) $D(0) \longleftarrow$ Calculate data placement solution with the lowest
(2) **for** $t = 1$ to T-1 **do**
(3) $D(t) \longleftarrow$ Find the solution with $\min(\text{TM}[D(t-1), :] + \text{SM}[:, t])$;
(4) **end for**
(5) $D^* = D[0: T-1]$
(6) **return** D^*;

Algorithm 4: Algorithm SOA: data placement optimization step by step.

Table 3: The total cost ($) of ADPA and SOA with varying data size in seven days (based on high DAF).

Data size (GB)	ADPA	SOA
100	1027.01	1042.33
200	2052.43	2082.52
300	3077.85	3122.71
400	4103.26	4162.90
500	5128.68	5203.09

Table 4: The total cost ($) of ADPA and SOA with varying data retrieval latency in seven days (based on high DAF).

Data retrieval latency (ms)	ADPA	SOA
200	6305.49	6370.93
300	5128.68	5203.95
400	5128.68	5203.95
500	5128.68	5203.95

Table 5: The total cost ($) of ADPA and SOA with varying data size within one year (based on low DAF).

Data size (GB)	ADPA	SOA
100	204.35	245.27
200	408.44	490.10
300	612.53	734.93
400	816.62	979.76
500	1020.71	1224.59

Table 6: The total cost ($) of ADPA and SOA with varying data retrieval latency within one year (based on low DAF).

Data retrieval latency (ms)	ADPA	SOA
200	204.35	245.27
300	164.65	188.16
400	126.42	157.22
500	126.42	157.21

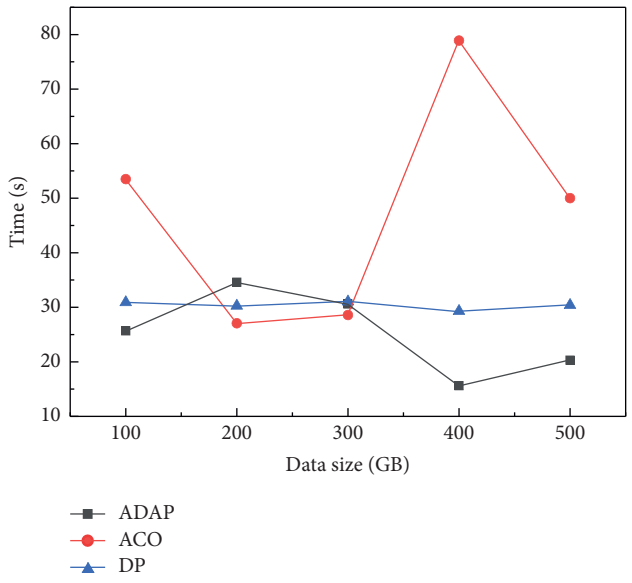

Figure 5: The comparison of run time.

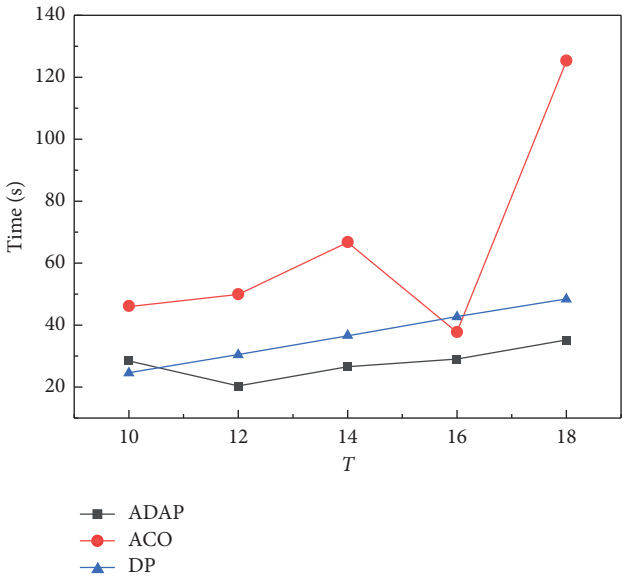

Figure 6: The comparison of run time.

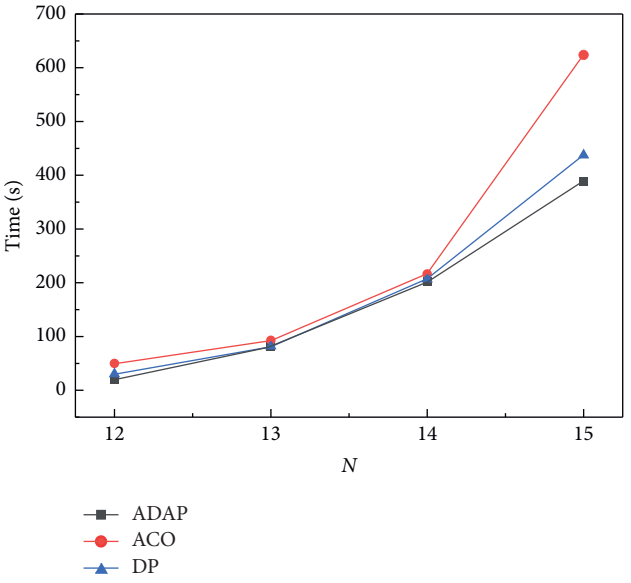

Figure 7: The comparison of run time.

Table 7: The total cost ($) comparison with varying data size within one year (based on low DAF).

Data size (GB)	ADPA	ACO	DP
100	204.35	238.51	238.51
200	408.44	476.73	476.73
300	612.53	714.94	714.94
400	816.62	953.16	953.16
500	1020.71	1191.38	1191.38

increases, the candidate solutions $D(t)$ have a clear growth trend. Assuming that data retrieval constraint is 500 ms, $T = 12$, data size is 500 GB and data availability is 0.999; $|D(t)|$ are 792, 1287, 2002, and 3003 when N varies from 12 to 15. When $N = 12$, 13, ADPA and ACO can solve the

Table 8: The total cost ($) with varying data retrieval latency within one year (based on low DAF).

Data retrieval latency (ms)	ADPA	ACO	DP
200	204.35	238.51	238.51
300	164.65	165.16	165.16
400	126.42	141.84	141.84
500	126.42	141.84	141.84

optimal solution, but when $N = 13, 14$, the cost of the ADPA and ACO is higher than the optimal scheme by \$0.99 and \$1.23, respectively. However, from Figure 7, as is manifested, the running time of ADPA is less than that of the other two algorithms in all N values; that is, the efficiency of solving the optimal solution is higher than that of ACO and DP. In this section, we also perform a comparative experiment on the dataset with lower DAF and achieve good results, as shown in Tables 7 and 8.

7. Conclusion

Users should adjust data placement solution based on DAF to minimize the storage, get operation, out-bandwidth, and migration costs in the whole life-cycle of data storage. In order to achieve this goal, we present an adaptive data placement architecture named *ADPA*. Because of the absence of the future DAF, the DAF prediction module based LSTM of ADPA can predict the future DAF through historical data. And then, the data placement optimization module, which is based on Q-learning of reinforcement learning, solves the optimal data placement solution sequence according to the prediction DAF. The experiments driven by a real workload of NASA-HTTP and cloud providers information indicate that ADPA not only is superior to the algorithm SOA but can save more time than ACO and DP to obtain the optimal data placement. In the future, we intend to present an architecture which can adjust data placement based on the change of cloud market.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

Pengwei Wang and Dong Wang are corresponding authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (NSFC) under Grant no. 61602109, DHU Distinguished Young Professor Program under Grant no. LZB2019003, Shanghai Science and Technology Innovation Action Plan under Grant no. 19511101802, Natural Science Foundation of Shanghai under Grant no. 19ZR1401900, Fundamental Research Funds for the Central Universities, and Alliance PlanSpecial Tenders For Difficult Problems under Grant no. LM201819.

References

[1] P. Wang, C. Zhao, and Z. Zhang, "An ant colony algorithm-based approach for cost-effective data hosting with high availability in multi-cloud environments," in *15th IEEE International Conference on Networking, Sensing and Control*, pp. 1–6, IEEE, Zhuhai, China, March 2018.

[2] P. Wang, C. Zhao, W. Liu, Z. Chen, and Z. Zhang, "Optimizing data placement for cost effective and high available multi-cloud storage," *Computing and Informatics*, vol. 39, no. 1, pp. 1001–1032. In press, 2020.

[3] Q. Zhang, S. Li, Z. Li, Y. Xing, Z. Yang, and Y. Dai, "Charm: a cost-efficient multi-cloud data hosting scheme with high availability," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 372–386, 2015.

[4] Y. Mansouri, A. N. Toosi, and R. Buyya, "Cost optimization for dynamic replication and migration of data in cloud data centers," *IEEE Transactions on Cloud Computing*, vol. 99, p. 1, 2017.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[7] Y. Mansouri, A. N. Toosi, and R. Buyya, "Data storage management in cloud environments: taxonomy, survey, and future directions," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, 2017.

[8] Y. Mansouri, A. N. Toosi, and R. Buyya, "Brokering algorithms for optimizing the availability and cost of cloud storage services," in *Proceedings of IEEE 5th International Conference on Cloud Computing Technology and Science*, pp. 581–589, Bristol, UK, December 2013.

[9] Y. Singh, F. Kandah, and W. Zhang, "A secured cost-effective multi-cloud storage in cloud computing," in *Proceedings of the 2011 IEEE Conference on Computer Communications Workshops*, pp. 619–624, Shanghai China, June 2011.

[10] M. Hadji, "Scalable and cost-efficient algorithms for reliable and distributed cloud storage," in *International Conference on Cloud Computing and Services Science*, pp. 15–37, Lisbon, Portugal, May 2015.

[11] Y. Qu and N. Xiong, "A resilient, fault-tolerant and high-efficient replication algorithm for distributed cloud storage," in *Proceedings of the 2012 41st International Conference on Parallel Processing (ICPP)*, pp. 520–529, Pittsburgh, PA, USA, September 2012.

[12] X. Liu, L. Fan, L. Wang, and S. Meng, "Multiobjective reliable cloud storage with its particle swarm optimization algorithm," *Mathematical Problems in Engineering*, vol. 2016, Article ID 9529526, 14 pages, 2016.

[13] M. Su, L. Zhang, Y. Wu, K. Chen, and K. LI, "Systematic data placement optimization in multi-cloud storage for complex requirements," *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1964–1977, 2016.

[14] N. K. Gill and S. Singh, "A dynamic, cost-aware, optimized data replication strategy for heterogeneous cloud data centers," *Future Generation Computer Systems*, vol. 65, pp. 10–32, 2016.

[15] X. Qiu, H. Li, C. Wu, Z. Li, and F. Lau, "Cost-minimizing dynamic migration of content distribution services into hybrid clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, 2015.
[16] T. S. Reddy and G. Murali, "Implementing the least-price cloud storage service in multiple cloud providers," in *Proceedings of the 2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, pp. 464–469, Coimbatore, India, October 2018.
[17] P. Wang, Y. Lei, P. R. Agbedanu, and Z. Zhang, "Makespan-driven workflow scheduling in clouds using immune-based PSO algorithm," *IEEE Access*, vol. 8, pp. 29281–29290, 2020.
[18] T. G. Papaioannou, N. Bonvin, and K. Aberer, "Scalia: an adaptive scheme for efficient multi-cloud storage," in *Proceedings of the 2012 International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–10, Salt Lake City, UT, USA, November 2012.
[19] C. Liu, X. Xu, and D. Hu, "Multiobjective reinforcement learning: a comprehensive overview," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 385–398, 2015.
[20] Amazon S3, 2018, https://aws.amazon.com/cn/s3/pricing/?nc=sn&loc=4.
[21] Microsoft Azure Cloud Storage, https://azure.microsoft.com/en-us/pricing/details/storage/.
[22] Alibaba Cloud Object Storage, https://www.aliyun.com/price/product/oss/detail.
[23] Google Cloud Storage, https://cloud.google.com/pricing/.
[24] 2018, http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html.
[25] P. Wang, W. Zhou, C. Zhao, and Y. Lei, "A dynamic programming-based approach for cloud instance types selection and optimization," *International Journal of Information Technology and Management*, vol. 19, no. 4, pp. 358–375, 2020.
[26] W. Liu, P. Wang, Y. Meng, Q. Zhao, C. Zhao, and Z. Zhang, "A novel model for optimizing selection of cloud instance types," *IEEE Access*, vol. 7, pp. 120508–120521, 2019.
[27] W. Liu, P. Wang, Y. Meng, G. Zou, and Z. Zhang, "A novel algorithm for optimizing selection of cloud instance types in multi-cloud environment," in *Proceedings of the 25th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2019)*, Tianjin, China, December 2019.
[28] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. Madhyastha, "Spanstore: cost-effective geo-replicated storage spanning multiple cloud services," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, (SOSP13), New York, USA*, p. 292308, 2013.
[29] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. Lau, "Scaling social media applications into geo-distributed clouds," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 8, pp. 689–702, 2015.
[30] A. Qureshi, *Power-demand routing in massive geo-distributed systems*, PhD thesis, MIT, Cambridge, MA, USA, 2010.
[31] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
[32] R. Sutton and A. Barto, *Reinforcement Learning: An introduction*, MIT press, Cambridge, MA, USA, 1998.
[33] W. Liu, P. Wang, Y. Meng, C. Zhao, and Z. Zhang, "Amazon EC2 spot instance price prediction using kNN regression," in *Proceedings of the 2018 Asia-Pacific Services Computing Conference, IEEE, Zhuhai, China*, January 2018.
[34] R. Bellman, "A Markovian decision process," *Indiana University Mathematics Journal*, vol. 6, no. 4, pp. 679–684, 1957.
[35] IBM Cloud Object Storage Pricing, https://www.ibm.com/cloud-computing/bluemix/pricing-object-storage.