Service Deployment Strategy for Customer Experience and Cost Optimization under Hybrid Network Computing Environment

Ning Wang[1*], Huiqing Wang[1], and Xiaoting Wang[1]
[1] School of Information Engineering, Shandong Management University, Jinan 250357, China
[e-mail: wangning200658@126.com]
*Corresponding author: Ning Wang

Received February 16, 2021; revised January 6, 2023; revised August 31, 2023; revised September 29, 2023; accepted October 18, 2023; published November 30, 2023

Abstract

With the development and wide application of hybrid network computing modes like cloud computing, edge computing and fog computing, the customer service requests and the collaborative optimization of various computing resources face huge challenges. Considering the characteristics of network environment resources, the optimized deployment of service resources is a feasible solution. So, in this paper, the optimal goals for deploying service resources are customer experience and service cost. The focus is on the system impact of deploying services on load, fault tolerance, service cost, and quality of service (QoS). Therefore, the alternate node filtering algorithm (ANF) and the adjustment factor of cost matrix are proposed in this paper to enhance the system service performance without changing the minimum total service cost, and corresponding theoretical proof has been provided. In addition, for improving the fault tolerance of system, the alternate node preference factor and algorithm (ANP) are presented, which can effectively reduce the probability of data copy loss, based on which an improved cost-efficient replica deployment strategy named ICERD is given. Finally, by simulating the random occurrence of cloud node failures in the experiments and comparing the ICERD strategy with representative strategies, it has been validated that the ICERD strategy proposed in this paper not only effectively reduces customer access latency, meets customers' QoS requests, and improves system service quality, but also maintains the load balancing of the entire system, reduces service cost, enhances system fault tolerance, which further confirm the effectiveness and reliability of the ICERD strategy.

Keywords: Service deployment, customer experience, cost optimization, fault tolerance

This work was supported by the National Natural Science Foundation of China (No. 61772068) and Shandong Social Science Planning Research Project (No. 19BJCJ18).

http://doi.org/10.3837/tiis.2023.11.007 **ISSN : 1976-7277**

1. Introduction

Advanced both in technology and concept, cloud can provide IT self-service, flexible resource expansion and pay-as-you-go IT service by combining mature IT infrastructure technologies with innovative service models. It also lays a foundation for the development of the mobile and networked information, and points out the development direction for computer technology of the next generation, which is highly anticipated by the industry. However, the import-and-export of data from cloud is indeed more complicated and difficult than people think. With the fast advancement of mobile Internet and IoT (Internet of Things), people are increasingly likely to rely on the cloud. As a result, there is a rising trend in the connectivity of intelligent devices to the Internet [1]. Howere, the amount of data generated every day not only brings huge transmission pressure to the network but also adds to the burden on data centers, so it is very unrealistic to transfer operations to the cloud for real-time data interaction at an almost TB level. As an extension and complement of cloud computing, fog computing and edge computing transmit data processing to data generation sources and try to reduce latency by cutting down the amount of data sent to the cloud. Although both edge computing and fog computing can reduce the system response time and improve system security, edge devices often have limited computing power and storage space, making it difficult to handle large-scale and complex computing tasks. Additionally, fog computing requires data to be transmitted to edge nodes for processing, which can increase network transmission costs if the data volume is large [2].

In the cloud, customers do not need to purchase their required software systems or configure corresponding hardware systems. They don't even need to provide maintenance personnel. Instead, they can simply lease the required services on-demand through the network. So, in cloud computing, the mode of multi-tenant sharing an application can reduce the operation and maintenance cost of the cloud service provider and the rental cost of customer. According to the tenant's requirement, the cloud service provider needs to determine the application instances to be deployed, the number of application instance replicas, and the cloud nodes to be used, etc. when their service platform is built completely. The issues need to be mainly considered when optimizing system resources. With the ongoing expansion of the cloud platform, the cloud service providers are facing some main problems like how to effectively deploy and manage applications, provide services for more tenants with the least resources and costs while obtaining maximum profits, and constantly improve the QoS requests for customers.

Unfortunately, node failures and unexpected outages are inevitable in cloud storage due to various unpredictable security risks, even the most reputable and the largest cloud vendors are not exempt from these challenges. As an example, in June 2018, Microsoft Azure experienced a downtime of over 5 hours[1]; In June 2019, Google Cloud Platform experienced a network outage for more than 4 hours, affecting the services in the western, eastern and central parts of United States[2]. The above examples show that cloud node failure has a serious impact on customer experience. Therefore, the fault-tolerant mechanism of cloud system is always a reliable guarantee for cloud services, which is a hot issue in cloud computing system management. At present, for cloud service providers, the problem of application recovery and

[1] http://www.myzaker.com/article/5c2bf94c77ac640156201ac5

[2] https://www.sdnlab.com/23850.html

redeployment after the failure of cloud nodes has become a research hotspot in this field, which has certain commercial and academic value.

The failure of cloud nodes is one of the main reasons affecting consumer experience, according to which this paper presents a strategy for optimizing resource deployment based on consumer experience and cost. The main contributions of this paper include:

(1) In the paper, the main factors affecting system load, including storage utilization, concurrent access rate, and bandwidth utilization, are considered. The paper proposes an alternate node filtering algorithm (ANF) to effectively ensure the quality of service.

(2) In the mathematical model of minimum service cost, it can be proved that the system performance can be optimized without changing the optimal solution when the adjustment factor of cost matrix is given.

(3) For the problem of data replica deployment, a "three ones" principle is proposed in this paper to avoid the phenomenon of the same data replica deployed to the same node as much as possible. So the probability of data replica loss is reduced and the fault tolerance of the system is improved. And the alternate node preference factor and alternate node preference algorithm ANP are also given in the paper. Based on these considerations, the CERD strategy is improved and optimized, resulting in the development of a new strategy called ICERD.

The experimental results demonstrate the effectiveness of the ICERD strategy in rapidly redistributing the application instances from the failed nodes to alternate nodes. The strategy keeps the system load balanced, reduces the service cost, and enhances the system fault tolerance.

Due to the mechanism of the service-popularity-determined number of the application instance replicas is adopted, and the concurrent access rate and bandwidth utilization rate of the cloud node are considered in the deployment, the customer's access latency is effectively shortened, the customers' QoS requests are effectively met, and the service quality of the system is therefore enhanced.

The subsequent sections of this paper are structured as follows: the related work and problem analysis are provided in Section 2. Section 3 defines the terminologies and symbols, and presents a mathematical model and its related properties. The model parameters are determined in this section, too. The details of ICERD are provided in Section 4 with theoretical analysis. In Section 5, the adaptive application program deployment strategy (ICERD) for cost optimization is proposed. In Section 6, some experimental analysis are conducted. And finally, the purpose of this paper is summarized, and the next possible work is discussed.

2. Related Work and Problem Analysis

2.1 Related Work

The future world will be an era in which everything is connected. With the improvement of technical standards and continuous breakthroughs of key technologies in IoT, the era of data explosion has come. If these huge amounts of data cannot be timely processed and utilized, they will soon become data garbage. How to solve the storage problem of mass data and process them effectively in real time are the key issues today. While providing a variety of data storage types, the cloud storage can offer a low-cost, infinitely scalable, highly reliable storage platform which is consists of not only thousands of servers, but also devices such as laptops, smartphones, tablets, sensors and smart routers, etc. However, these devices may fail at any time. Therefore, for cloud service providers, improving redundant replica and failure recovery mechanisms are the guarantee to achieve reliable storage of applications in the cluster. The recovery of application replica and the deployment of all applications are directly related to the service cost and the quality of cloud storage, which are also a hotspot issue in the field of cloud storage.

In recent times, there has been a significant surge in research related to cloud storage systems and the management of their replicas. The implementation of an optimal number of replicas is crucial for reducing storage costs in a cloud storage system. However, the conventional 3-replica replication strategy often leads to excessive storage space consumption, thereby incurring unnecessary expenses [3]. In the prior research, the focus was on matching the number of data backups with the specific requirements of clients. To achieve this, the minimum cost of storage blocks (MCSB) strategy was proposed, which effectively met the storage needs of users while minimizing costs. And the system load was considered and adaptive MCSB (AMCSB) strategy was presented in order to ensure the optimal performance of the system [4]. Li et al. examined the trade-off between storage cost and data reliability. By implementing proactive tracking and detection mechanisms, they were able to effectively reduce the number of data copies, leading to significant cost savings in storage. However, it is important to note that this approach also introduced additional system overhead due to the continuous detection process [5]. Bao et al. conducted a study aiming to decrease the repair time by minimizing repair costs. they focused on the encoding and repair methods of cross-datacenter erasure codes and proposed a hybrid-structured repair method, in [6]. [7] studied the cost minimization of online virtual network function (VNF) backup in the edge computing environment, proposed the Drift-Plus-Penalty online backup deployment scheme.

In cloud computing, when failure nodes are identified, it is common practice to replicate and redeploy the missing data to ensure data recovery and maintain data reliability. In [8], utilizing an enhanced secret sharing algorithm, Zhang et al. proposed a method for distributed data backup and recovery, which could quickly recover data on the backup controller in the event of a failure, greatly improving network availability. [9] proposed a two-stage container failure recovery strategy considering application priority, and used formal methods and greedy heuristics to deal with the failure recovery of high priority and low priority applications respectively, so as to improve the failure recovery efficiency of container applications in the edge computing environment. Lin et al. proposed a novel failure prediction technique in [10]. The technique allows for the identification of potentially failure-prone nodes in cloud by analyzing historical data, even before the actual occurrence of node failures. Gupta et al. combined an effective fault-tolerant method with an encryption algorithm to select the minimum fault-tolerant nodes [11]. In [12], Chinnathambi et al. gave a simple and reliable Byzantine fault detection method, and designed a scheduling algorithm and checkpoint optimization algorithm on this basis, which could be fault-tolerant and eliminated before the impact of Byzantine faults. In [13], a method to automatically provide delay-aware failover strategy through server placement algorithm was proposed. According to the failure probability of file, [14] presented the minimum replicas formula of reaching file expected availability, and realized a high scalable dynamic replicas management strategy in cloud storage system. In [15], Wang et al. presented a technique of selective data recovery in the event of node failure, taking into account service cost, service quality and load, and gave an economic and effective replica deployment strategy (CERD). [16] focused on online failure prediction for cloud computing and presented a failure prediction method, which leveraged the Hidden Markov Model and Cloud Theory to achieve an optimal balance between failure prediction accuracy and computational overhead. [17] introduced a multi-class load balancing method aimed at distributing various classes of customer tasks across multiple heterogeneous computing nodes. The objective of this method was to minimize the per-class mean response time. In cloud, the issues of updating cloud node, adding data and modifying data greatly affected the system load, [18] designed a resource allocation mechanism based on dynamic pricing and task allocation algorithms to achieve effective load balancing of cloud services and enhance the utilization of cloud.

The providers face a challenging task of efficiently handling dynamic customer service requests while ensuring cost-effectiveness and adherence to service level agreement (SLA). For minimizing the overall execution time of priority tasks, policy iteration scheduling (PIS), proposed by [19], offered a novel approach to globally optimize strategy of independent task scheduling. Mohan et al. proposed an optimized framework for placing blocks in geographically distributed storage clusters to minimize the average cost of single block repair, so as to achieve a significant improvement in repair performance in case of node failure [20]. In [21], Xia et al. studied quality-of-service (QoS)-aware data replication and placement and offered an efficient heuristic algorithm. As for continuously supporting QoS requirements, [22] proposed a random diffusion search algorithm that could minimize the replication cost of data. In order to meet the requirements of QoS based on SLA, resource management strategies must be considered. A resource scheduling model consisting of two levels was proposed by [23], with the goal of decreasing service delay and enhancing task execution stability.

Table 1. Comparison of resource deployment algorithms in cloud computing

Reference	Methodology	Advantage	Disadvantage
[3]	The conventional 3-replica replication strategy in Hadoop	• Fault tolerance • High availability • Load balancing	• Increase storage overhead • Increase network overhead • Not considering cost
[4]	The adaptive minimum cost of storage blocks (AMCSB) strategy	• Load balancing • Reduce service cost • Improve service quality	• Not considering bandwidth • Not considering concurrent tasks
[5]	A novel cost-effective reliability management mechanism	• High availability • Optimization storage • Reduce service cost	• Not considering system load • Not considering data access performance issues
[9]	Double applause recovery mechanism characterized by application priority	• Fault tolerance • Reduce system latency	• Not considering cost • Not considering system load
[14]	A cost-effective dynamic replication management scheme	• Optimization storage • Load balancing	• The cost is not the lowest • Not considering bandwidth • Not considering concurrent tasks
[15]	A cost-effective replica deployment strategy	• Load balancing • Improve service quality • Reduce service cost	• Not further optimizing system performance • Poor fault tolerance
[18]	A resource allocation mechanism based on dynamic pricing	• Load balancing • Improve cloud resource utilization	• Not considering cost • Not considering bandwidth • Not considering concurrent tasks

From the above related work, it can be seen that the current influencing factors of the whole system for the deployment of service resources in the network environment are not comprehensively considered, and less attention is given to users' requirements. In this paper, customer experience is emphasized as the focal point, with service cost as the optimal goal. The focus is on the systemic impact of deploying services on load, fault tolerance, and service cost.

2.2 Problem Analysis

In the network computing mode, optimizing deployment problem is to determine the deployment relationship and resource allocation between the tenants and the application instances, the application instances and the servers, so the system resource utilization rate and the software supplier's profit can be maximized, and the service cost can also be minimized while tenant's service requests are met. In order to improve the availability of the cloud platform and the system fault tolerance, the replica mechanism of the application instance is widely used. Even if the cloud node fails in the cloud platform, by automatically acquiring copies from other fully functional cloud nodes, the system ensured the availability and reliability of the cloud platform.

Because the huge cloud storage systems are composed of various cloud nodes with different performance, and the most of these machines are cheap and unreliable. Furthermore, considering the unreliable network connections and limited bandwidth, the failure of cloud nodes is considered a routine occurrence rather than an exceptional event. Since any data stored on the failure node will no longer be valid, the failure of the cloud node will inevitably cause some data replicas to be lost or lower than the specified value, which will seriously affect the quality of users' service requests. Therefore, the system will continuously detect the data that needs to be replicated and redeploy them to other cloud nodes to achieve the specified number of replicas and meet the user's QoS requests. So as to ensure the efficiency and availability of the system.

In a large-scale data storage system, each cloud node differs in storage capacity and acceptable concurrent access. When the cloud node's access capacity is saturated, a request from a new client will be blocked or rejected. Thus, cloud service providers have been pursuing the goal of reducing the service cost of storage data as much as possible without affecting the system utilization and reliability.

Firstly, the storage capacity available sets inherent limitations on the amount of data that can be stored. Additionally, as the number of replicas increases, the management of the system becomes increasingly complex. Moreover, the inclusion of additional replicas incurs additional management cost, ultimately leading to higher service cost for the cloud platform. For real-time application instances, generally, they can only be visited frequently by tenants in a certain period of time. Therefore, only by adopting a real-time and reasonable application instance replicas mechanism and storing all application instances and their replicas on cloud nodes in a distributed way, via parallel data transmission, service performance and transmission efficiency can be enhanced by considering the factors mentioned above and calculating the optimal number of replicas based on the popularity of each service [4].

At present, although cloud computing can reliably store data with high fault tolerance and provide high throughput data access, the nodes in cloud computing are generally composed of low-cost nodes, so hardware errors are normal rather than abnormal. Faced with this situation, as shown in **Fig. 1**, the main problems studied in this paper focus on the basic services of IaaS (Infrastructure as a Service) to ensure service cost and service quality. Firstly, how to obtain the lost data information and its popularity. Secondly, how to select the alternative nodes in the cloud, which will affect the system service quality; finally, how to redeploy these data fastly and efficiently, maintain system load balance, decrease service costs, improve service quality and enhance system fault tolerance.

On this basis, a strict mathematical model for the aforementioned problems is established by introducing the corresponding terms and symbols, and the existence of optimization strategies for these problems is discussed. The problem can be simplified as follows: determining the number of data replicas that can ensure a good customer experience and considering the available storage space, distribution, and storage cost in the cloud storage system. Furthermore, a cost-optimized redeployment mechanism is designed and implemented for lost data in the event of node failure.

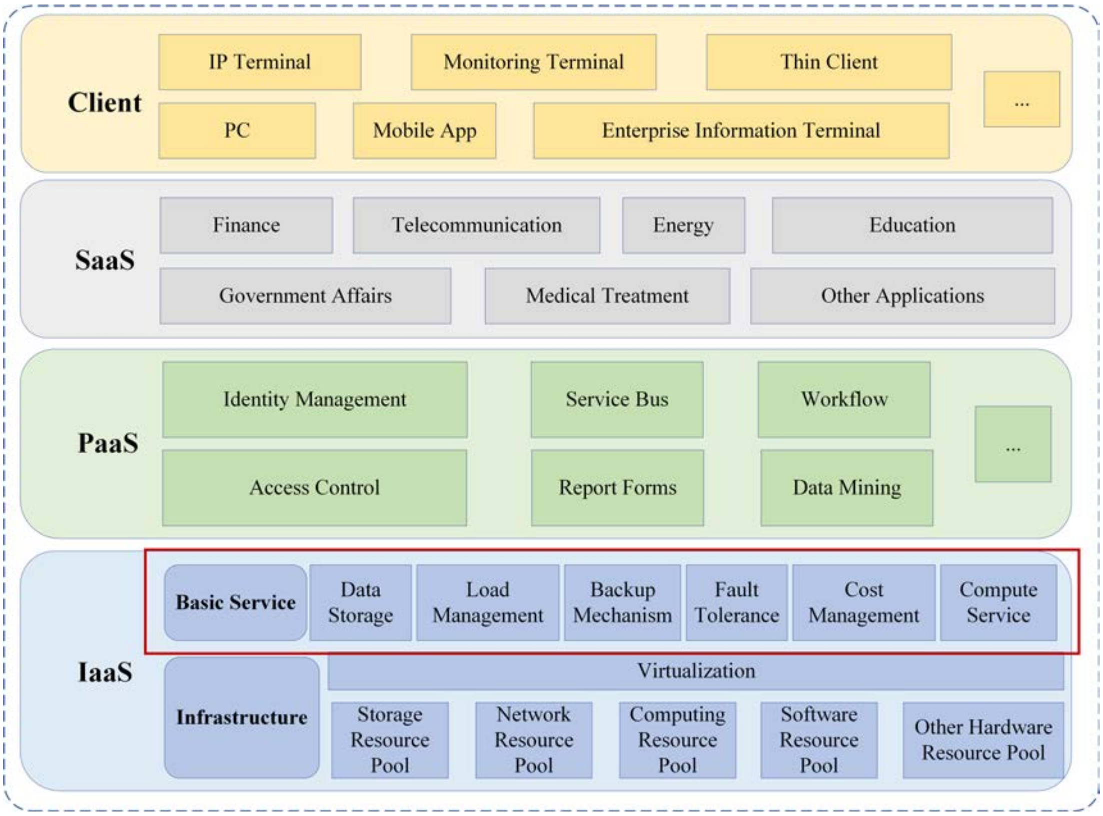

Fig. 1. Cloud computing architecture

3. Mathematical Model

3.1 Terminology and Definition

To enhance the clarity and understanding of the problem discussed in this paper, let us define certain terms and principles:

(1) Total number of nodes in the network computing resources: Denoted as Y. This represents the total count of computing resources available in the network.

(2) Number of failed nodes: Denoted as F ($F \leq Y$). It refers to the count of nodes that have experienced failure.

(3) Number of data blocks stored in the failed nodes: Denoted as *B*. This indicates the number of data blocks that are currently stored in the failed nodes.

Additionally, the definitions of service cost, service total cost, data block's popularity, and data deployment principle are given as follows:

Definition 1: The service cost c_{ij} represents the cost of utilizing the *i*-th cloud node to store the *j*-th data block. The comprehensive cost of storing all data blocks in cloud nodes is referred to as the overall service cost.

Definition 2: In a specific time period, data block's popularity (hot_j) refers to the proportion of accesses made to the *j*-th data block out of the total number of accesses made to all data blocks.

Definition 3: This paper adopts the principle that two or more replicas of the same data block cannot be deployed to the same node in one resource deployment. this principle is called the "three ones".

3.2. Alternate Node Filtering Method

In this paper, because the main aim is to optimize system performance, improve system fault tolerance, reduce the storage cost and improve the customers' QoS requests, when analyzing the system load, they are considered comprehensively which are the real-time availability of storage space, concurrent access and bandwidth utilization of cloud nodes, described as follows:

(1) Storage space

In order to describe the disk space utilization of cloud nodes, the storage utilization of cloud node D_k ($1 \le k \le D$) is represented by L_k ($1 \le k \le D$), which is computed by:

$$L_k = {u_k}/{s_k} \tag{1}$$

Where u_k represents the occupied storage capacity of cloud node D_k, and s_k represents the total storage capacity of cloud node D_k. $0 \le L_k \le 1$, when $L_k=0$, it means that the cloud node D_k is not used and is in idle state currently.

Let's define *SumD* as the total number of cloud nodes in the system, and *SL* as the system's disk space utilization. *SL* is computed by:

$$SL = \sum_{k=1}^{D} L_k \Big/ SumD \tag{2}$$

(2) Concurrent access

Set the concurrent access rate T_k of a cloud node, which is defined as:

$$T_k = \frac{Task_k}{SumTask_k} \tag{3}$$

In Eq. (3), $Task_k$ represents the number of the tasks currently being executed by the *k*-th cloud node, and $SumTask_k$ represents the maximum number of the tasks that can be executed by the *k*-th cloud node at the same time.

The overall concurrent access rate of the system is *CA*, which is computed by:

$$CA = \sum_{k=1}^{D} T_k \Big/ SumD \tag{4}$$

(3) Bandwidth utilization

Set the bandwidth utilization of the cloud node W_k, as shown in Eq. (5):

$$W_k = \frac{w_k}{SumW_k} \tag{5}$$

In Eq. (5), w_k represents the amount of bandwidth occupied by node D_k, $SumW_k$ represents the total bandwidth allocated to cloud node D_k.

The overall bandwidth utilization rate of the system is BW, which is computed by:

$$BW = \sum_{k=1}^{D} \mathrm{W}_k \Big/ SumD \tag{6}$$

Then, Eq. (7) describes the load condition LB_k of cloud node D_k.

$$LB_k = L_k \times W_{sl} + T_k \times W_{ca} + W_k \times W_{bw} \tag{7}$$

$$W_{sl} + W_{ca} + W_{bw} = 1$$

$$0 \le L_k \le 1, 0 \le T_k \le 1, 0 \le W_k \le 1, \min(L_k, T_k, W_k) \le LB \le \max(L_k, T_k, W_k)$$

Where W_{sl}, W_{ca} and W_{bw} are the weights of disk space utilization, concurrent access and bandwidth utilization of cloud nodes.

The load of the system is shown in Eq. (8), this value is called the load threshold LB in this paper.

$$LB = SL \times W_{sl} + CA \times W_{ca} + BW \times W_{bw} \tag{8}$$

$$W_{sl} + W_{ca} + W_{bw} = 1$$

$$0 \le SL \le 1, 0 \le CA \le 1, 0 \le BW \le 1, \min(SL, CA, BW) \le LB \le \max(SL, CA, BW)$$

When $0 \le LB_k \le LB$, the node load belongs to the cloud node set L_{light} of light load, otherwise, it is categorized as the cloud node set L_{heavy} of heavy load. For the data that need to be redeployed, the alternative nodes are the elements in the set L_{light}, with the total number $D=|L_{light}|$. The alternate node filtering algorithm (ANF) is described in **Algorithm 1**:

Algorithm 1. Alternate node filtering

Input: storage space parameters, concurrent access parameters, and bandwidth utilization parameters of cloud nodes

Output: alternate nodes' labels and total number D

Step 1: According to Eq. (1), the storage utilization of each node is calculated respectively;

Step 2: According to Eq. (2), the storage space utilization of the system is calculated;

Step 3: According to Eq. (3), the concurrent access rate of each node is calculated;

Step 4: According to Eq. (4), the overall concurrent access rate of the system is calculated;

Step 5: According to Eq. (5), the bandwidth utilization of each node is calculated;

Step 6: According to Eq. (6), the overall bandwidth utilization of the system is calculated;

Step 7: Use Eq. (7) to calculate the load of each cloud node LB_k;

Step 8: Use Eq. (8) to calculate the whole system load LB;

Step 9: When $0 \le LB_k \le LB$, the node load belongs to the cloud node set L_{light} of light load, otherwise, it is categorized as the cloud node set L_{heavy} of heavy load;

Step 10: $D=|L_{light}|$, output alternate nodes' labels and total number D.

3.3 Mathematical Model

Generally, due to the low failure rate of cloud nodes, the total number of data blocks lost is relatively small. According to the "three ones" principle, and considering the load of cloud node and the advantages of distributed parallel computing, when the nodes fail, the lost *B* data blocks are backed up again, and data are redeployed on *D* cloud nodes.

The strategy proposed in this paper will be considered in the following two cases: (1) when *B*≤*D*, it will be transformed into a classical assignment problem to solve until all the data are deployed in the nodes. (2) When *B*>*D*, if continuing to use the assignment method, it will increase the probability that the same node stores multiple replicas of the same data, and reduce the fault-tolerant ability of the system. In this paper, the local assignment algorithm is used to resolve the problem, which is described in Section 5.2 of this paper, until all the data are deployed in the nodes.

Suppose that the total number of nodes equals the total number of data blocks after adjustment, which is represented by *B*. Because the service cost of cloud nodes is different, the purpose of storing *B* data blocks in *B* cloud nodes is to decrease system service cost, improve the service quality and enhance the fault tolerance of the system. In other words, our problem has been reformulated as an optimization problem, as depicted in Eq. (9).

$$\min(S) = \sum_{j=1}^{B}\sum_{i=1}^{B} c_{ij}x_{ij} \tag{9}$$

$$s.t.\begin{cases} x_{i1} + x_{i2} + \cdots + x_{ij} + \cdots + x_{iB} = 1 \\ \qquad i = 1,2,\cdots,B \\ x_{1j} + x_{2j} + \cdots + x_{ij} + \cdots + x_{Bj} = 1 \\ \qquad j = 1,2,\cdots,B \\ x_{ij} = 0,1; i = 1,2,\cdots,B; j = 1,2,\cdots,B \end{cases}$$

This is an integer programming of type 0-1. Its matrix form is shown in Eq. (10):

$$\min(S) = CX \tag{10}$$

$$s.t.\begin{cases} AX = 1 \\ x_{ij} = 0\text{或}1, i, j = 1,2,...,B \end{cases}$$

Where *A* is 2*B*×*B*^{2} matrix.

$$X = (x_{11}, x_{12}, ..., x_{1B}, x_{21}, x_{22}, ..., x_{1B}, ..., x_{B1}, x_{B2}, ..., x_{BB})^T,$$

$$C = (c_{11}, c_{12}, ..., c_{1B}, c_{21}, c_{22}, ..., c_{1B}, ..., c_{B1}, c_{B2}, ..., c_{BB})$$

$$X_{ij} = \begin{cases} 1: \text{The jth block is stored in the ith node；} \\ 0: \text{The jth block is not stored in the ith node；} \end{cases} (1 \le i \le B,\ 1 \le j \le B)$$

Then, the matrix *C* of the service cost is depicted in Eq. (11):

$$\boldsymbol{C} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1B} \\ c_{21} & c_{22} & \cdots & c_{2B} \\ \vdots & \vdots & \vdots & \vdots \\ c_{B1} & c_{B2} & \cdots & c_{BB} \end{bmatrix} \ge 0 \tag{11}$$

Theorem 1: In any column or any row of service cost matrix *C*, subtracting or adding a constant will not impact on the optimal solution of the problem.

Currently, take subtracting a constant from any row as an example to prove that other cases are similar.

Proof. After subtracting a constant h from any row in the service quality cost matrix for adjustment, the problem is transformed into the optimal solution for $\min(S)^{'}$. Based on the conditions stated in Eq. (9), $\min(S)_{=}\min(S)^{'}+h$ can be got. Therefore, the optimal solution will not be affected after the adjustment. In other words, the service cost will not be affected.

If there are N 0 elements in the matrix C of service cost that are located in different columns of different rows, then the optimal solution of the problem is to take 1 as the corresponding variables to these 0 elements and 0 as the rest of the variables. The basic idea of our strategy is to subtract a certain constant from all the rows columns of service cost matrix C, and then convert C into N 0 elements which are located in different columns and different rows, so that the variables corresponding to these 0 elements are taken as 1, and other variables are taken as 0, then the optimal solution of the problem is obtained.

4. Determination of Model Parameters

4.1 Service Coset c_{ij}

In the cloud, according to their business needs, users usually rent corresponding services from the cloud providers. The service cost c_{ij} is determined by factors including software and hardware, such as the number of CPUs, their cores and main frequency, the size of memory, operating system (Linux and windows), hard disk and network disk, database (SQL server, my SQL, Oracle, PostgreSQL, Cassandra and MongoDB), network bandwidth (Mbps), as well as the maintenance software cost and hardware cost, and the geographical location of rental services. The factors listed above are all in direct proportion to the rental cost. The rental unit price typically follows a calculation based on different time intervals, such as hours, months, or years. The popular cloud service providers like Amazon, Alibaba, and Grandcloud determine their prices based on the previous factors. This paper takes the cloud quotation of Grandcloud as the parameter, and carries out the relevant experimental test work.

4.2 Data Block's Popularity and Number of Replicas

In the experiment section of our paper, the popularity of replicas is utilized to predict the number of replicas for the big data samples [4].

4.3 Service Quality

The quality of service provided by cloud platform for users can be measured in the following three aspects: (1) the rapidity of time, which mainly refers to whether the service provider is sensitive to users′ requests and swift in action, which means the shorter the time, the better. (2) Whether the technology is standard, that is to say whether the service provider′s technology is actually excellent and it can really solve problems for customers. (3) Whether the commitment is reliable, that is, whether the service provider and the service requester can fulfill the service level agreement (SLA). In this paper, the average service response time of the system is mainly used as a measure of service quality.

5. Optimization Strategy

5.1 Adjustment Factor of Cost Matrix

Definition 4 The adjustment factor E_{ij} of cost matrix reflects the relationship between the service cost of the same data node and the popularity of the stored data block, which is the extra cost caused by the popularity. Let E_{ij}=min(c_{ij})×hot_i, where min (c_{ij}) is the minimum value of all service costs and hot_i is the popularity of data block B_i. If the popularity of data is not considered, then $hot_i = 0$.

The cost matrix is adjusted according to Eq. (12). Through simple analysis, it can be seen that all nodes expect to serve the data with lower popularity, which can reduce the number of concurrent tasks of the node, shorten the service delay and improve the service quality.

$$C_{ij} = c_{ij} + E_{ij} \tag{12}$$
$$i = 1, 2, ..., D;\ j = 1, 2, ..., B;$$

According to Theorem 1, it can be ensured that the optimal solution of the strategy remains unchanged after the cost adjustment factor is increased, but the phenomenon of unbalanced resource allocation can be skillfully avoided, so as to improve the access efficiency of users. The service costs mentioned below refer to C_{ij} adjusted by Eq. (12).

In fact, using the adjustment factor of cost matrix is an optimization strategy. According to theorem 1, the optimization strategy can enhance the system service performance without affecting the minimum total service cost of the storage data block. In this paper, the service cost parameters used in the strategy refer to the price C_{ij} adjusted by the adjustment factor of cost matrix unless otherwise stated.

5.2 Optimization Strategy

As this paper mainly aims at optimizing system performance, improving the fault-tolerant ability of the system, reducing the service cost of data storage and improving the QoS request of users, and according to the "three ones" principle, if $B \le D$, it can be solved by assignment algorithm; if $B > D$, it puts forward a new solution strategy, which is described as follows:

(1) All data blocks are sequenced in a descending manner according to popularity;

(2) Set $n=\lceil B/D \rceil$, then $m= B-n \times D$ ($0 \le m < D$);

(3) For the previous $n \times D$ data blocks, according to the "three ones" principle, that is, in D nodes, the assignment algorithm is used to deploy D data blocks at one time, and after n times of calling the assignment algorithm, the number of undeployed data blocks is m.

(4) For these m data blocks, a mathematical model for selecting alternative nodes is offered, as shown in Eq. (13):

$$A_{ki} = C_{ki} \times w_1 + LB_k \times w_2 \tag{13}$$
$$（k = 1, 2, ..., D; i = 1, 2, ..., m; w_1 + w_2 = 1)$$

Where C_{ki} is the service cost adjusted by Eq. (12). LB_k is the load condition of cloud node D_k. w_1 and w_2 refer to the weights of service cost and node load respectively. When the value of A_{ki} is smaller, it shows that using D_k to store the i-th data block can effectively decrease the service cost and enhance the system service quality. In this paper, A_{ki} is called the alternate node preference factor.

(5) $m \times D$ A_{ki}（$1 \le k \le D, 1 \le i \le m$）are divided into m groups according to i, and A_{ki} of each group is sequenced ascendingly to form a set A_i of optional node optimization factors. （$i=1,2,\ldots,m$）

(6) Judge the elements in the set A_i in turn. The deployment strategy of data block B_i is to select the minimum A_{ki} from the set A_i, and if the same data block B_i is not stored on node D_k, then deploy data block B_i to D_k. And so on, until all the nodes in the m sets are deployed to the corresponding nodes.

The alternate node preference algorithm (ANP) is shown in **Algorithm 2**:

Algorithm 2. Alternate node preference

```
Input: $B$ and $D$
Output: all nodes are deployed to the corresponding nodes
Step 1: Sort $B$ data blocks by decreasing popularity;
Step 2: Set $n$=[$B$/$D$], then $m$= $B$-$n$×$D$(0≤$m$<$D$), the data blocks are
        divided into $n$+1 group. There're $D$ data blocks in the first $n$
        groups each, and $m$ data blocks in the last group ($m$<$D$);
Step 3: for( i=1; i≤n; i++ )
            Call (CERD1 ($C$)) [15] ;// Deploy $D$ data blocks in $i$-th group in
            turn
Step 4: For $m$ data blocks in group n+1, $A_{ki}$ is calculated by Eq. (13),
        (1≤$k$≤$D$,1≤$i$≤$m$);
Step 5: $A_{ki}$ is divided into $m$ groups according to $i$. The $A_{ki}$ of each group
        is sequenced in an ascending way to form the set $A_i$ of alternative
        node preference factors ($i$=1,2,…,$m$);
Step 6: Judge the elements in set $A_i$ in turn. If the same data block $B_i$ is
        not stored on node $j$ ($j$=1,2,…,$D$), deploy the data block $B_i$ to $D_j$.
        Otherwise, judge the next node $D_{j+1}$. And so on, until all nodes
        are deployed to the corresponding nodes.
```

According to this, the paper introduces an improved CERD strategy, namely ICERD, which can decrease the service cost and enhance the system fault tolerance as shown in **Strategy 1**.

The core steps of ICERD strategy are explained as follows:

Step 1: Subtract the minimum element of each row from the elements of that row, and the minimum element of each column from the elements of that column in cost matrix C to ensure there will be as many 0 elements as possible in cost matrix C.

Step 2: Start from the row or column with the fewest 0 elements, find a 0 element, make the identifier of the element be M, the identifier of all other elements in the row and column where the 0 element is located is N, and so on. If the number of elements whose identifier is M in matrix C is D, the optimal solution is obtained. If the number of elements whose identifier is M in matrix C is less than D, then go to step 3.

The step 3 is completed in the following five parts:

(1) Set the identifiers of all rows and columns in the matrix to be P;

(2) If the identifier of all elements of a certain row in the matrix is N, then the row identifier of that row is O;

(3) If the row contains 0 elements, then the column identifier of the column where the 0 element is located is O;

(4) The row identifier corresponding to all 0 elements in the column is O;

(5) Repeat part (3) and part (4) of step 3 until no more row identifier or column identifier is obtained;

Step 4: Find the minimum value among all the elements with the row labels as O and the column labels as P. Following that, add the minimum value to all elements with the column labels as O, and subtract the minimum value among all elements with the row labels as O. Return to step 2.

If B≤D, then CERD1 algorithm is called for solution [15], otherwise call ANP algorithm for solution.

Strategy 1. Improved cost-efficient replica deployment

```
1. Obtain the lost data information from the system, and calculate the
   number of data that needs to be recovered $B$;
2. Call ANF algorithm to find out the information and number of
   spare nodes $D$;
3. To adjust the service cost $C$ by Eq. (12);
4. If ($B$=$D$) {Execute CERD1($C$);} // reference paper [14]
5. If ($B$<$D$)
     { In cost matrix $C$, to append ($D$-$B$) blocks, where the associated
         service costs are all set to 0.;
       Execute CERD1($C$);
       break;
     }//reference paper [15]
6. If ($B$>$D$) then {Call (ANP( ));}
7. For each element in set $X$, if $x_{ij}$=1, it indicates that the $j$-th block
   will be stored in $i$-th node;
8. The minimum total cost of service t is computed by Eq. (9);
9. the solution is output.
```

6. Experiment Analysis

The cloud nodes in our cloud test platform are composed of 50 representative servers selected from hundreds of servers. They are distributed in different provinces of the country with different machine performances. In the experiments, the configuration and quotation of these cloud nodes refer to the cloud quotation of Grandcloud's East China nodes[3], as shown in **Table 2**. To address the problem, the approach of random direct power failure is adopted to simulate the failure of cloud nodes during the experiments. However, the number of failed cloud nodes is small in actual cloud environment. Therefore, the failure rate F is 2%, 4% and 6% respectively. The number of data (128M) with different popularity degrees lost is about 360-5760, and the average response time for user requests at a certain time is tested from 1000-4000. Among them, the weights of storage utilization, concurrent access and bandwidth utilization of cloud nodes are W_{sl}=W_{ca}=33%, W_{bw}=34%, and the weights of service cost and node load are w_1=w_2=50% respectively.

Table 2. The configuration and price of nodes

	D_1(5)	**D_2(9)**	**D_3(12)**	**D_4(12)**	**D_5(6)**	**D_6(6)**
Configuration	1core\1G\15G	1core\2G\30G	1core\4G\60G	2core\8G\120G	4core\16G\240G	8core\32G\480G
OS	CentOS7.6	CentOS7.6	CentOS7.6	CentOS7.6	CentOS7.6	CentOS7.6
Java Version	1.8.0_291	1.8.0_291	1.8.0_291	1.8.0_291	1.8.0_291	1.8.0_291
Hadoop Version	3.3.6	3.3.6	3.3.6	3.3.6	3.3.6	3.3.6
Service cost (¥/day)	4.1	7.9	16.1	31.9	64.1	128.2

[3] http://www.grandcloud.cn/

Firstly, when the failure rate of cloud node is *F*=4%, the data blocks are backed up in the cloud node with random power failure, and test the ICERD strategy proposed in this paper in HDFS. Under the assumption of the same cloud node failures, the distribution of data recovered is compared with Hadoop default strategy [3], CDRM strategy [14], PRCR strategy [5] and CERD strategy [15] respectively. In the end, the storage utilization of each cloud node in the system is compared as shown in **Fig. 2**. It can be found that the ICERD strategy and the CERD strategy can both effectively redeploy data when the cloud nodes fail, and make the load of the system closer to the balance, and the values are relatively smooth. Although the CDRM strategy is not specifically targeted for the failure of cloud nodes, it mainly adopts a dynamic adjustment strategy, but the load factor is also considered. Therefore, the load effect of the system is relatively good and the effect of Hadoop default strategy is ordinary. PRCR strategy mainly considers the reliability of data replica and compression of storage space and does not consider the load of the whole system, so the values fluctuate greatly.

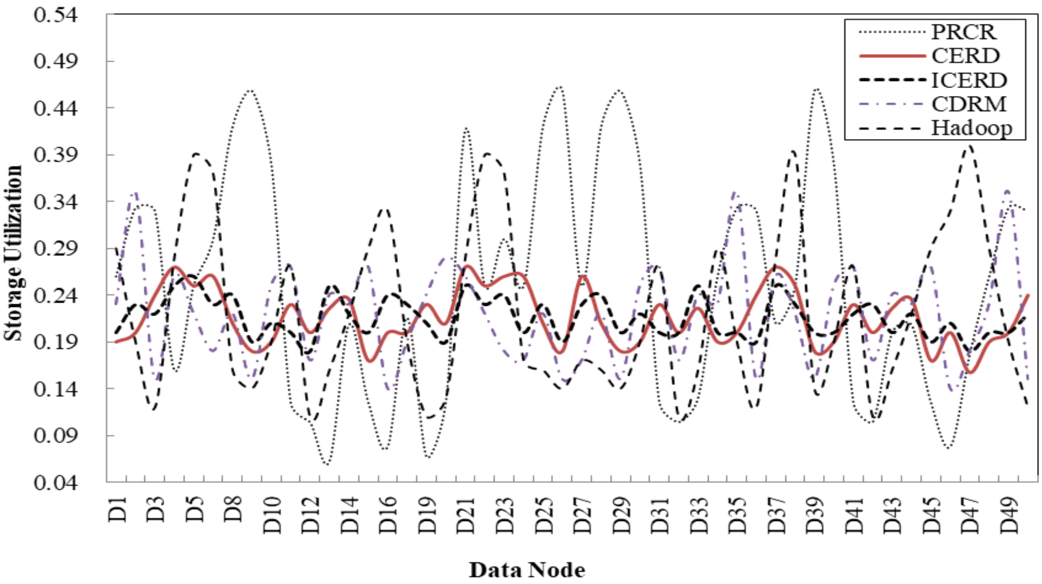

Fig. 2. *F* = 4%, after data is backed up and redeployed, the storage utilization of each node.

Secondly, the comprehensive analysis is conducted on how different strategies respond to changes in requests. Based on the previous experiment results, the analysis is conducted on utilizing the average response time as a measurable standard for QoS. To optimize the system service quality, the service cost used in this paper are C_{ij} which is adjusted by the cost matrix adjustment factor. Considering 1000-4000 concurrent user requests, the average response time of the system is tested and the experiment results are presented in **Fig. 3**. Because the ICERD strategy proposed in this paper considers not only the disk space utilization and concurrent task execution of each cloud node, but also the bandwidth utilization. At the same time, the adjustment factor of cost matrix is also proposed in this paper, which avoids the situation that the replicas are too centralized in the nodes with low service cost and effectively improves the service performance of the system. Therefore, the experiment results are shown that the average service response time of the ICERD strategy proposed in this paper is shorter than that of CERD strategy by 7.4%, CDRM strategy by 12.9%, Hadoop strategy by 24.6% and PRCR strategy by 21.2%. In the experiment, for verifying that the matrix cost adjustment factor proposed has the certain optimization effect on the system performance, the ICERD strategy parameters are the unadjusted service cost c_{ij} and the adjusted service cost C_{ij} respectively. As shown in **Fig. 4**, the experiment result of ICERD (C_{ij}) is better than that of ICERD (c_{ij}) by 2.5%.

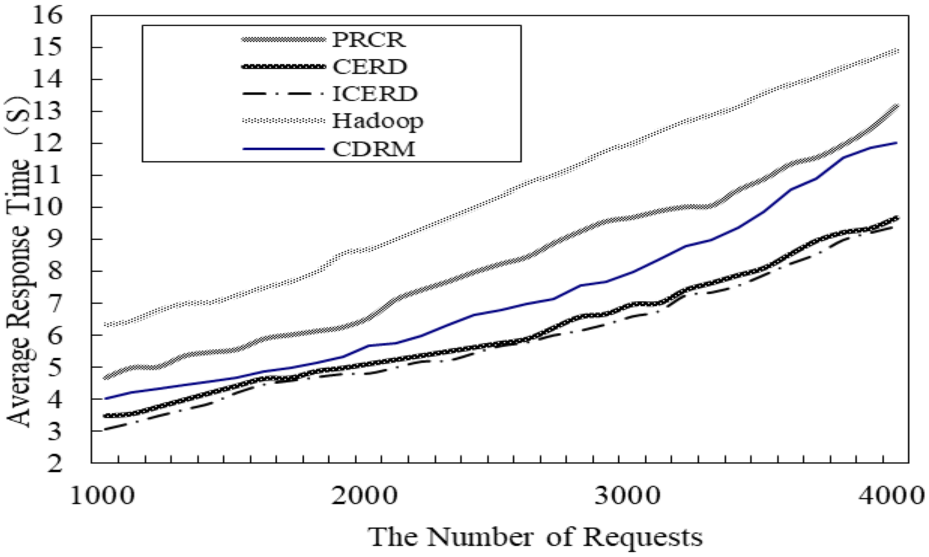

Fig. 3. F = 4%, after data is backed up and redeployed, the average response time of various strategies along with the growing of the increasing number of requests.

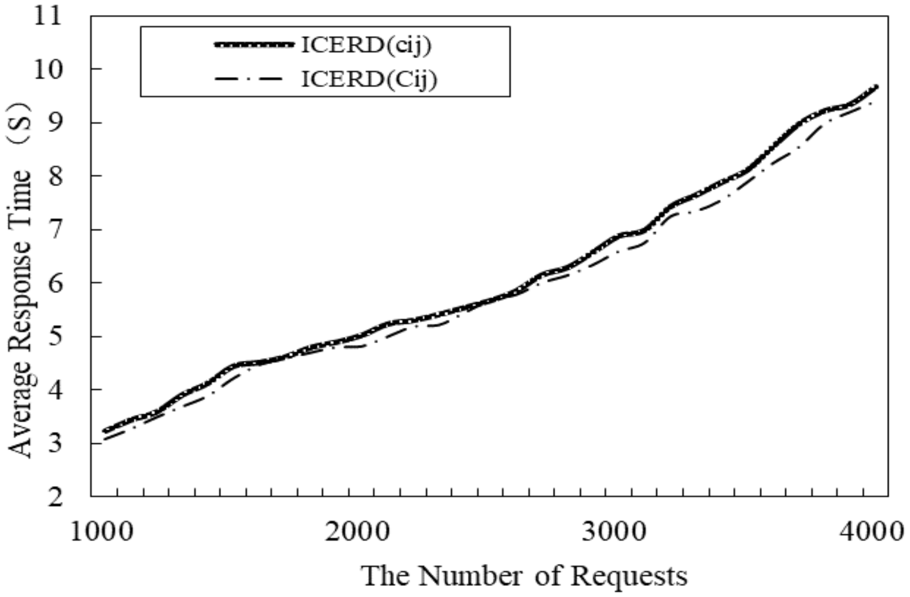

Fig. 4. F = 4%, following the data backup and redeployment with the growing of the number of requests, the strategy parameters are the unadjusted cost c_{ij} and the adjusted cost C_{ij} separately and then the average service response time is tested and compared.

Thirdly, for verifying the cost optimality of ICERD strategy, the total service cost of various strategies and the experiment results are presented in **Fig. 5**. Finally, when the cloud node failure rate *F* is 2%, 4% and 6% respectively, the service performance of the system is tested for 1000 concurrent requests and accessing more popular data at the same time as presented in **Fig. 6**. When $B \leq D$, the total service cost of ICERD strategy and CERD strategy is the lowest and when $B > D$, from **Fig. 5** and **Fig. 6**, it can be found that the total service cost of ICERD strategy is lower. Although the cost of ICERD strategy is slightly higher than that of CERD strategy by 1.7%, it effectively reduces the probability of the same data block replica being deployed in the same node and enhances the system fault tolerance with a lower cost, so the experiment results align with the theoretical analysis. Since the main consideration of PRCR strategy is to compress storage space and reduce service cost, the total service cost is relatively low. But the CDRM strategy and Hadoop strategy mainly consider the system load, so the total cost of service is relatively high. Therefore, the ICERD strategy can better meet the users' service requests and improve the fault tolerance of the system, which is better than the other four strategies.

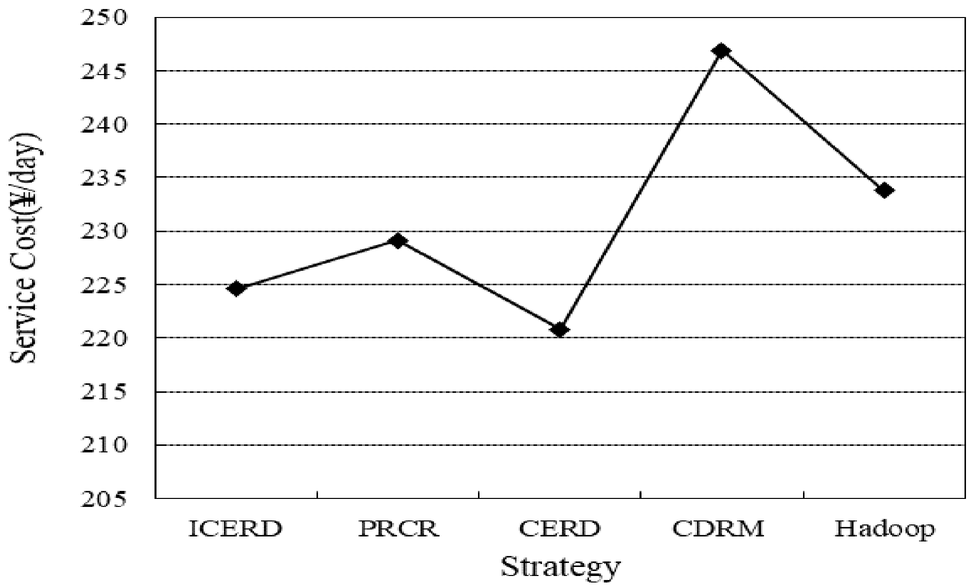

Fig. 5. *F* = 4%, following the data backup and redeployment, the total system service cost with various strategies.

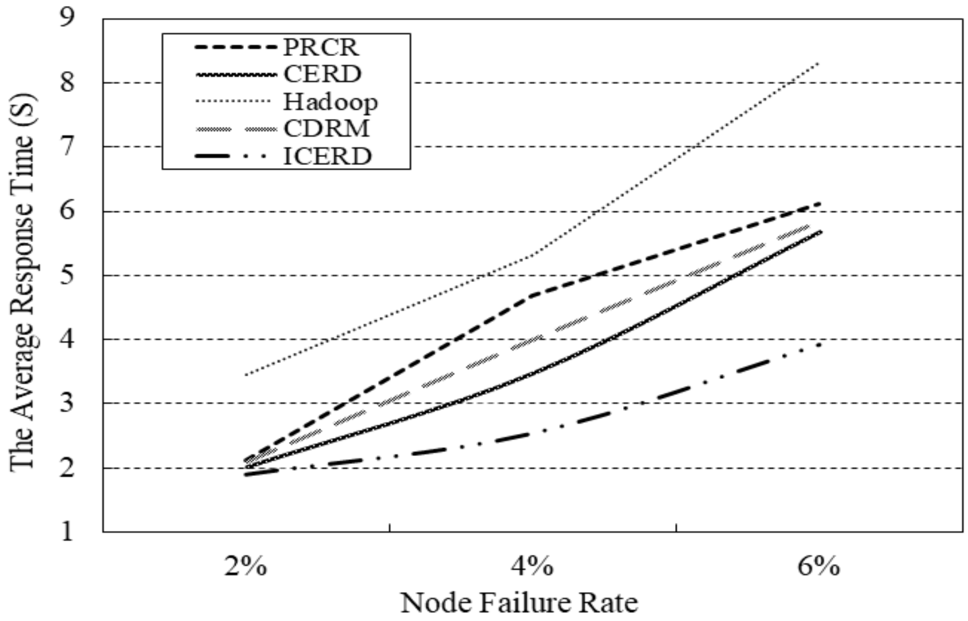

Fig. 6. *F* is 2%, 4% and 6% respectively. After data is backed up and redeployed, when 1000 requests access the data with higher popularity at the same time, the average response time of different strategies.

7. Conclusion

More and more applications are supported by distributed computing resources in network environment extensively. To solve the problem of optimization strategy for service deployment, a service resource deployment method based on service request popularity is offered, which can better guarantee the optimization of customer experience and service cost. Briefly, by obtaining the relevant information of the failed cloud node through NameNode, the main factors are first considered that affect the system load, including storage utilization, concurrent access, and bandwidth utilization. The ANF algorithm which is proposed in this paper can effectively guarantee the system service quality.

Secondly, in order to decrease the system service cost, the paper proposes an adjustment factor of cost matrix based on the minimum service cost mathematical model, which can optimize the system performance without changing the optimal solution, and gives its proof.

Finally, the alternate node preference factor and ANP algorithm are proposed to avoid the phenomenon of the same data backup being deployed to the same node, which reduces the loss probability of data replicas, improves the fault tolerance of the system. And the ICERD strategy is developed, which is based on cost and load optimization.

By comparing it with the other four strategies in the experiments, it is verified that while reducing the service cost, the ICERD strategy proposed in this paper is conducive to the system load balance, and able to improve the service performance and fault tolerance of the system. However, there is a certain gap between the simulated environment and the real-time big data scenario. In response to the dynamic network computing environment and the growing user demands, the optimal cost deployment strategy is the main content of the following research under the dynamic change of network resources.

References

[1] M. Marjani, F. Nasaruddin, A. Gani, A.Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, "Big IoT data analytics: architecture, opportunities, and open resea rch challenges," *IEEE Access*, vol 5, pp. 5247-5261, 2017. Article (CrossRef Link).

[2] Y. Hajjaji, W. Boulila, I. R. Farah, I. Romdhani, and A. Hussain, "Big data and IoT-based applications in smart environments: A systematic review," *Computer Science Review*, vol 39, pp. 1-39, 2021. Article (CrossRef Link).

[3] H. T. Almansouri and Y. Masmoudi, "Hadoop Distributed File System for Big data analysis," in *Proc. of 2019 4th World Conference on Complex Systems (WCCS)*, pp. 1-5, 2019. Article (CrossRef Link).

[4] N. Wang, Y. Yang, K. Meng, Y. Chen, and L. Wang, "A Customer Experience-Based Cost Mini mization Strategy of Storing Data in Cloud Computing," *Acta Electronica Sinica*, vol. 42, pp. 20-27, 2014. Article (CrossRef Link).

[5] W. Li, Y. Yang, J. Chen, and D. Yuan, "A cost-effective mechanism for cloud data reliability management based on proactive replica checking," in *Proc. of 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pp. 564-571, 2012. Article (CrossRef Link).

[6] H. Bao, Y. Wang, and F. Xu, "Reducing network cost of data repair in erasure-coded cross-datacenter storage," *Future Generation Computer Systems*, vol. 102, pp. 494-506, 2020. Article (CrossRef Link).

[7] Y. Liu, X. Shang, and Y. Mao, "Availability Aware Online Virtual Network Function Backup in Edge Environments," *IEEE Transactions on Mobile Computing*, vol. 2023, pp. 1-14, 2023. Article (CrossRef Link).

[8] Y. Zhang, L. Zhong, and S. Yang, "Distributed data backup and recovery for software-defined wide area network controllers," *Transactions on Emerging Telecommunications Technologies*, vol. 33, pp. 1-17, 2022. Article (CrossRef Link).

[9] K. Ray and A. Banerjee, "Prioritized fault recovery strategies for multi-access edge computing using probabilistic model checking," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, pp. 797-812, 2023. Article (CrossRef Link).

[10] Q. Lin, K. Hsieh, Y. Dang, H. Zhang, K. Sui, Y. Xu, J. Lou, C. Li, Y. Wu, R. Yao, M. Chintalapati, and D. Zhang, "Predicting Node failure in cloud service systems," in *Proc. of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 480-490, 2018. Article (CrossRef Link).

[11] V. Gupta, B. P. Kaur, and S. Jangra, "An efficient method for fault tolerance in cloud environment using encryption and classification," *Soft Computing*, vol. 23, pp. 13591-13602, 2019. Article (CrossRef Link).

[12] S. Chinnathambi, A. Santhanam, J. Rajarathinam, and M. Senthilkumar, "Scheduling and checkpointing optimization algorithm for Byzantine fault tolerance in cloud clusters," *Cluster Computing*, vol. 22, pp. 14637-14650, 2019. Article (CrossRef Link).

[13] Y. Aldwyan and R. O. Sinnott, "Latency-aware failover strategies for containerized web applications in distributed clouds," *Future Generation Computer Systems*, vol. 101, pp. 1081-1095, 2019. Article (CrossRef Link).

[14] Q. Wei, B. Veeravalli, B. Gong, L. Zeng, and D. Feng, "CDRM: A cost-effective dynamic replication management scheme for cloud storage cluster," in *Proc. of 2010 IEEE international conference on cluster computing*, pp. 188-196, 2010. Article (CrossRef Link).

[15] N. Wang, Y. Yang, Z. Mi, Q. Ji, and K. Meng, "A fault-tolerant strategy of redeploying the lost replicas in cloud," in *Proc. of 2014 IEEE 8th International Symposium on Service Oriented System Engineering*, pp. 370-375, 2014. Article (CrossRef Link).

[16] W. Zheng, Z. Wang, H. Huang, L. Meng, and X. Qiu, "EHMM-CT: An Online Method for Failure Prediction in Cloud Computing Systems," *KSII Transactions on Internet & Information Systems*, vol.10, pp. 4087-4107, 2016. Article (CrossRef Link).

[17] S. F. El-Zoghdy, A. Ghoneim, "A Multi-Class Task Scheduling Strategy for Heterogeneous Distributed Computing Systems," *Ksii Transactions on Internet & Information Systems*, vol.10, pp. 117-135, 2016. Article (CrossRef Link).

[18] C. Kumar, S. Marston, R. Sen, and A. Narisetty, "Greening the cloud: a load balancing mechanism to optimize cloud computing networks," *Journal of Management Information Systems*, vol. 39, pp. 513-541, 2022. Article (CrossRef Link).

[19] B. Hu, N. Xie, T. Zhao, and X. Zhang, "Dynamic Task Scheduling Via Policy Iteration Scheduling Approach for Cloud Computing," *Ksii Transactions on Internet & Information Systems*, vol.11, pp. 1265-1278, 2017. Article (CrossRef Link).

[20] L. J. Mohan, K. Rajawat, U. Parampalli, and A. Harwood, "Optimal placement for repair-efficient erasure codes in geo-diverse storage centres," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 101-113, 2020. Article (CrossRef Link).

[21] Q. Xia, Z. Xu, W. Liang, S. Yu, S. Guo, and A. Y. Zomaya, "Efficient Data Placement and Replication for QoS-Aware Approximate Query Evaluation of Big Data Analytics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, pp. 2677-2691, 2019. Article (CrossRef Link).

[22] C. Li, C. Wang, H. Tang, and Y. Luo, "Scalable and dynamic replica consistency maintenance for edge-cloud system," *Future Generation Computer Systems*, vol. 101, pp. 590-604, 2019. Article (CrossRef Link).

[23] X. Ma, H. Gao, H. Xu, and M. Bian, "An IoT-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, p. 249, 2019. Article (CrossRef Link).

Ning Wang, born in Yantai City, Shandong Province in 1978, professor, received her B.E. degree in computer science and technology from Shandong University of Technology, Zibo, China in 2003, the M.Sc. degree in computer software and theory from Yunnan Normal University, Kunming, China in 2006, and the Ph.D. degree in communication and information system, University of Science and Technology Beijing, Beijing, China in 2015. Currently she is teaching in the school of information engineering, Shandong Management University and her research interests include cloud computing and big data analysis.

Huiqing Wang, born in Jinan City, Shandong Province in 1982, associate professor, received her B.E. degree in mathematics and applied mathematics from Qufu Normal University, Qufu, China in 2005, the M.Sc. degree in probability theory from Qufu Normal University, Qufu, China in 2008, and the Ph.D. degree in management science and engineering from Southeast University, Nanjing, China in 2015. Currently she is teaching in the School of Economics and Trade, Shandong Management University and her research interests include corporate finance and financial management. Huiqing Wang is regarded as the joint first author.

Xiaoting Wang, born in Jinan City, Shandong Province in 1990, lecturer, received her B.E. degree in computer science and technology from Shandong University of Science and Technology, Qingdao, China in 2013, the M.Sc. degree in web technology from University of Southampton, U.K in 2014. Currently she is teaching in the school of information engineering, Shandong Management University and her research interests include semantic analysis and big data analysis.