Model-Based Evaluation and Optimization of Dependability for Edge Computing Systems

Jingyu Liang, Bowen Ma, Sikandar Ali, and Jiwei Huang(✉)

Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum - Beijing, Beijing 102249, China
2020310701@student.cup.edu.cn, {sikandar,huangjw}@cup.edu.cn

Abstract. Edge computing moves part of the computing tasks to the edge of the network to improve service capabilities while reducing latency. It has been successfully applied in Internet of Things (IoT) and mobile computing systems. With the increasing popularity of edge computing, the ability of an edge computing system continuously providing services to users without interruptions and failures, which is also known as the dependability, has become an important issue. However, the evaluation and optimization of dependability attributes of an edge computing system still remains an largely unexplored problem. In this paper, we study this issue from a model-based viewpoint. We propose an atomic dependability model of a server and provide quantitative analyses of dependability attributes with Markov chain techniques. In order to facilitate the analyses of multiple attributes in large-scale environments, we adopt a state aggregation method for model simplification, and present its corresponding theoretical proof. Considering the edge-cloud collaboration, we put forward the dependability model of an edge computing system, and provide an evaluation approach using the state aggregation technique. Furthermore, taking task offloading as an example, we formulate the dependability optimization as a continuous-time Markov decision problem (CTMDP), and propose an efficient approach of solving the problem with reinforcement learning. Finally, we use a real-world dataset to conduct simulation experiments, and the experimental results validate the efficacy of our approach.

Keywords: Dependability · Edge computing · Continue-time Markov decision process · State aggregation

1 Introduction

Cloud computing, as a business model that provides services on demand through the network, has been widely applied in IT industry [1,2]. However, with the emergence of mobile smart devices and wireless communication technologies such as Internet of Things (IoT) and 5G, the requirements for low latency have raised

H. Gao and X. Wang (Eds.): CollaborateCom 2021, LNICST 406, pp. 728–747, 2021.
https://doi.org/10.1007/978-3-030-92635-9_42

dramatically, which brings significant challenges to cloud computing. Especially for date-intensive services, a extremely large amount of data needs to be uploaded from users to the cloud, which costs plenty of time for data transmission through the Internet. To attack this challenge, edge computing as a novel computing paradigm has been proposed. Servers are deployed at the edges of the networks very close to the terminal devices, and they are able to provide services directly to users for reducing latency by avoiding all service requests being uploaded to the cloud [3,4]. Also, with edge computing, the bandwidth pressure within the backbone network as well as energy consumption within the cloud can be reduced.

Meanwhile, the computational capability of an edge server is quite limited comparing with a cloud data center, and thus some computation-intensive tasks still need to be handled by the cloud. Instead of replacing the cloud, edge computing needs to cooperate with cloud computing to better fulfill user requests [5,6]. For example, in factories, edge servers can process data generated by equipment more quickly and detect abnormal situations in time. The edge nodes periodically upload the processed data to the cloud for storage and management. At the same time, the cloud is also responsible for monitoring data transmission. With the close cooperation between the edge and the cloud, the factories can be operated and managed in a very high efficiency. Within such a complex multi-layered architecture, how to design an advanced task offloading scheme has become an urgently required issue to be addressed.

As the IoT and mobile computing have been widely applied in various scenarios especially for some critical applications such as medical health-care, electric power management, military applications, etc. Besides high performance, dependability is also one of the most important requirements, which is the degree of stability to which a system can provide fault-free services without interruption [7]. In order to guarantee the dependability of a computing system, its evaluation and optimization is a critical research problem. Although there have been several works dedicating to study the dependability issue in traditional HPC systems and cloud computing by analyzing system log data or constructing mathematical models [8], the dependability research of systems with close collaboration between edge computing and cloud computing has still not been explored to a large extent.

In this paper, we make an attempt at filling this gap by studying the dependability evaluation and optimization for edge computing systems from a mathematical modeling aspect. By analyzing large-scale system failure data and surveying existing work, we propose a state transition model of edge computing systems. With detailed mathematical analyses, the dependability attributes can be quantitatively evaluated. A state aggregation approach is designed, with which a comprehensive system model of an edge-cloud collaboration system can be established and theoretically analyzed with very high efficiency. Furthermore, we formulate the dependability optimization for task offloading in edge computing as a continuous-time Markov decision process (CTMDP), and introduce reinforcement learning techniques to solve the problem efficiently. Finally, we conduct simulation experiments based on failure data collected from real-life computing systems, and validate the effectiveness of our approach.

The remainder of this paper is organized as follows. In Sect. 2, we survey the related work most pertinent to this paper. In Sect. 3, the dependability state transition model of the edge and the cloud is established, and dependability attributes are analyzed and quantified. In Sect. 4, a state aggregation technique is put forward, with which the dependability model of an edge-cloud collaboration system is proposed. In Sect. 5, the dependability optimization problem is formulated by a continuous-time Markov decision process. To solve such problem with high efficiency in large-scale systems, the time difference method and reinforcement learning are introduced and algorithms are presented. In Sect. 6, we conduct simulation experiments based on real-life dataset, and validate our approach. Finally, Sect. 7 concludes this paper.

2 Related Work

Research on dependability has always been of profound significance. Pan et al. provided a general definition of cloud computing system dependability, and discussed some methods for establishing cloud computing reliability models [9]. Guan *et al.* proposed a cloud dependability analysis (CDA) framework and designed a failure metric directed a cyclic graph (DAG) to analyze the correlation between various performance indicators and failure events in virtualized and non-virtualized systems [10]. In [11], authors research and set goals to connect different kinds of servers with private clouds and designate the servers as hosts to establish a multi-cloud dependability system. Qiu *et al.* considered parallel redundant big data tasks, and proposed a joint modeling method to consider the correlation between reliability-performance and reliability-energy consumption during failures, and proposed a genetic algorithm to find the optimal solution [12]. In [13], firstly according to the priority of the service requested by the user, the use of redundant resources is analyzed from the perspective of dependability index and service cost. Then the reliability of the proposal received from the cloud provider was evaluated based on the negotiation strategy suggested for the requester. Bai *et al.* proposed a method for constructing a complex network model that regards cloud services as nodes and finds the relationships between services to construct a complex network model. Through relationships key services can be found in a large number of services, thereby improving cloud services reliability [14]. Zhang *et al.* proposed an optimization model based on queuing theory. Using this optimization model, energy can be minimized under the constraints of average response time, response time reliability, stability of the queuing system, and resource maximization. Optimal resource allocation for cloud computing strategies [15]. Luo *et al.* proposed a resource scheduling algorithm that can optimize reliability, system performance, and energy consumption based on the Semi-Markov model modeling method. Furthermore, Laplace-Stieltjes transform and Bayesian method are used to analyze the correlation between reliability and performance and reliability and Energy [16]. In [17], the author proposes a mechanism that can continuously update the reliability of cloud resources and provide users with reliable resource scheduling in the cloud computing environment. In [18], the author provides a newest solution to automate the negotiation process in the cloud environment.

With the rapid growth of user service requests, many people have begun to research edge computing. For instance Song *et al.* [19] proposed regular assignment of tasks in the edge computing in order to increase the number of tasks that can be processed in the edge computing network and at the same time to ensure efficient task management. Some main technologies that can support the edge paradigm are introduced in [20]. Further, authors discussed some existing problems and proposed relevant solutions along with the advantages and precautions of each solutions in different scenarios [20]. An edge computing architecture to overcome the network bottleneck caused by the massive increase in the internet of Things (IoT) devices and data with the development of cloud computing are proposed in [21]. Their architecture is based on the based on the $\lambda - CoAP$ architecture, covering edge computing and cloud infrastructure. Regarding the placement of edge servers in an edge computing environment, Chen *et al.* [22], proposed an effective method of placing edge servers to reduce access latency, optimize load balance, and prevent the big data generated at the edge from making the cloud computing center unable to process effectively. In [23], a risk-based mobile edge computing system resource configuration optimization method is proposed. The method is capable to tackle the risk of edge server overload while performing resource allocation, and further improve system efficiency and service quality. Xiao *et al.* proposed a predictive mapping optimization heuristic method to place edge servers to meet the resource demand forecasting problem, divide tasks into subtasks, map the positions of servers and subtasks, and complete the information interaction between server and data, and proposed a cross-regional resource optimization model to reduce costs [24].

Although numerous published articles address [13,19–22], various Edge/Cloud computing subjects such as design and architectures [19], optimal resource allocation strategies [13], resource configuration optimization [21], resource demand forecasting [22], and task offloading [23]. However, the dependability of edge computing and cloud collaboration is not considered in none of the above work, which is one of the important factors to ensure the normal service provided by the system.

3 Dependability Modeling and Analysis of Edge/Cloud Server

This section builds a state model based on the service computing system, and introduces dependability and its attributes. In order to analyze and quantify these attributes, consider discrete-time Markov Chain (DTMC) for embedding.

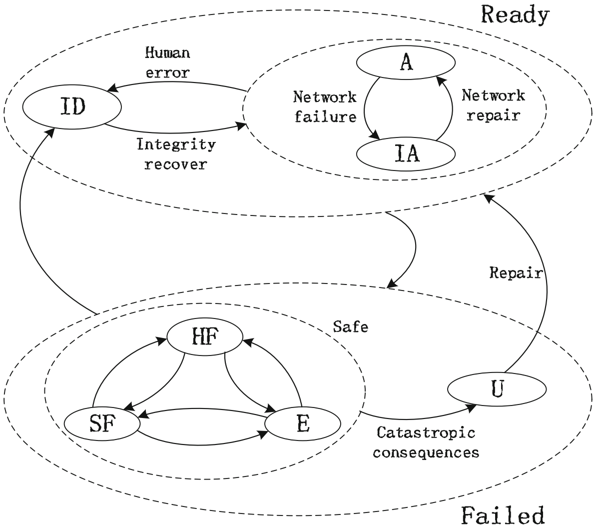

Fig. 1. Service computing system dependability mode

3.1 System State Transition Model

During the operation of the system, there might be a series of problems, such as failure, recovery and so on. Since this series of events will affect the state of the whole service system and have a decisive impact on the processing capacity of the whole system. It is important to discuss the state transition of the system according to the failure in the service computing system for the evaluation of dependability [25]. In this paper, in addition to the analysis of the types of failures, the correlation between failures within the system is also considered, that is, some failures of the system will affect the system, and lead to new failures before the recovery of the failure, which is called the correlation of failure.

According to a large-scale study of failures in high-performance computing systems [8], the fault is roughly divided into Human error; Environment; Network failure; Software failure; Hardware failure. In our previous work, we studied the state transition transition of the service computing system [26]. On this basis, we considered the correlation between faults, and considered the correlation between different faults to help us further analyze the dependability, the system dependability model is shown in Fig. 1. The status is hierarchical, divided into two parts as *Ready* or *Failed*. In the ready state to further divide the system, whether the system has a state can be divided into the Integrity to ensure Integrity-Maintained and Integrity-Destroyed (ID) two kinds of state. Integrity under the guarantee of Accessible (A) on behalf of all the components in the system are normal state, but in the process of system processes the request, may be due to failure of the network system can't receive the user request, unable to complete the user request, this state is defined as Inaccessible (IA). Integrity-Destroyed state means that when the system is processing a service request, the system may be modified by inappropriate modules or data due to human error, but this change has not caused system failure or service failure.

In this paper, referring to the failed state of large-scale service systems, hardware failures and software failures account for a large proportion. Therefore, hardware failures and software failures are the biggest causes of failures, but environmental failures are also very important. We regard these types of failures as repairable failures and classify them as Safe. Unsafe (U) means that the failure may lead to catastrophic consequences. In different situations, users or service providers are required to define the level of catastrophic consequences.

The CTMC has a transition probability matrix P_{ij}, and uses the transfer rate of the CTMC to calculate its transfer probability, as

$$P_{ij} = \frac{q_{i,j}}{\sum_{k \neq i} q_{i,k}} \cdot (1 - e^{-\sum_{k \neq i} q_{i,k} \xi}), i \neq j \tag{1}$$

$$P_{ij} = e^{-\sum_{k \neq i} q_{i,k} \xi}, i = j \tag{2}$$

By solving a system of linear equations, steady-state probability $v = [v_{ID}, v_A, v_{IA}, v_{HF}, v_{SF}, v_E, v_U]$ embedded a discrete-time Markov chain (DTMC) in each state.

$$\begin{cases} v \cdot P = \pi \\ \sum_i v_i = 1 \end{cases} \tag{3}$$

Assuming that the average waiting time of each state in the CTMC model is t_i and the steady-state probability of DTMC is π_i, the steady-state probability of any state $i \in \{ID, IA, A, HF, SF, E, U\}$ in the CTMC model can be obtained as V:

$$\pi_i = \frac{v_i t_i}{\sum_j v_j t_j}, \forall\ i. \tag{4}$$

As it has been assumed that each epoch has time period of equal length $t_i = \xi$ for any i, it is obvious that

$$\pi_i = \frac{v_i t_i}{\sum_i v_j t_j} = \frac{v_i}{\sum_i v_j} = v_i. \tag{5}$$

Therefore, the embedded DTMC steady-state probability and the original CTMC steady-state probability are equal. The expected DTMC reward in steady-state is expressed ,which is proved equivalent to the CTMC reward function shown in

$$\bar{R} = \sum_{i \subseteq s} = v_i \cdot r_i = \pi_i \cdot r_i \tag{6}$$

3.2 Analysis of Dependability Attributes

This paper introduces a widely recognized dependability, which is an aggregate concept that contains five attributes. Based on the definition of dependability attribute and the proposed state model of service computation, this paper makes a mathematical analysis of some attributes of dependability attribute, so as to evaluate the service system more directly [27].

From the perspective of system design and optimization, generally speaking, steady-state dependability analysis is more meaningful than transient analysis (that is, to analyze the dependability attributes of the system at a certain time t). In view of aforementioned definition of dependability attributes, based on the steady-state probability of states, quantitative analysis of different attributes can be carried out.

Availability represents the ability of the system to provide a working service to the user when the user needs it, and is expressed as:

$$A = Pr(Accessible(t)) = \pi_A \tag{7}$$

Integrity reflects the ability of the service computing system to resist inappropriate systems, use the following expression to analyze the attribute.

$$I = \pi_A + \pi_{IA} \tag{8}$$

Reliability is the probability that services can work continuously at a given time. In steady-state, reliability is the steady-state probability of the ready state, and is expressed as:

$$R = \pi_A + \pi_{IA} + \pi_{ID} \tag{9}$$

Safety provides services without catastrophic consequences, it can be expressed as follows:

$$S = \pi_{ID} + \pi_A + \pi_{IA} + \pi_{SF} + \pi_{HF} + \pi_E \tag{10}$$

Maintainability describes the ability to repair and modify the system. This article considers that all faults can be repaired, so the expression of Safety and Maintainability in this paper are the same.

4 Dependability Modeling and Analysis of Edge Computing System

To simplify the complex system model, in this section, a state aggregation method is proposed, and further prove that the models before and after aggregation are equivalent. Additionally, on the basis of this an edge-cloud collaboration model is proposed.

4.1 State Aggregation Technique

In the service computing system, due to the large number of servers and complex distribution, it is difficult to directly obtain a complete model, the solution is to aggregate complex models. The purpose of state aggregation is to reduce the state space and aggregate it into a two-state Markov reward model. The reward values of Ready (R) state and Failed (F) state are set to 1 and 0 respectively. The aggregation model is shown in Fig. 2, here the red transition (RT) is define as a state transition from the ready state to the failed state, while the transition from the failed state to the ready state as green transition (GT) [28].

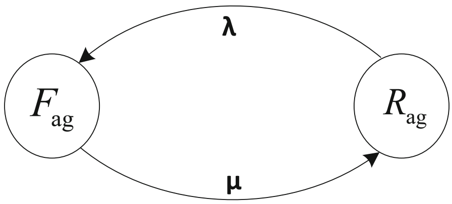

Fig. 2. The aggregated model

The transition rate from up to down state is set as λ and is denoted as:

$$\lambda = \sum_{t_{i,j} \subseteq RT} Pr\{s_i \mid R\} \cdot q_{i,j} = \frac{\sum\limits_{t_{i,j} \subseteq RT} \pi_i q_{i,j}}{\pi_{RT}} \tag{11}$$

$$\pi_{RT} = \sum_{s_k \subseteq R} \pi_k \tag{12}$$

s_i is the state i, π_k is the CTMC steady-state probability in state k, $t_{i,j}$ shows the transition from state i to state j. μ is similar:

$$\mu = \sum_{t_{i,j} \subseteq GT} Pr\{s_i \mid F\} \cdot q_{i,j} = \frac{\sum\limits_{t_{i,j} \subseteq GT} \pi_i q_{i,j}}{\pi_F} \tag{13}$$

$$\pi_F = \sum_{s_k \subseteq F} \pi_k \tag{14}$$

The aggregation model expressed in $\pi = [\pi_{R_{ag}}, \pi_{F_{ag}}]$ steady-state probability [29], and the steady-state probabilities are:

$$\pi_{R_{ag}} = \frac{\mu}{\lambda + \mu} \tag{15}$$

$$\pi_{F_{ag}} = \frac{\lambda}{\lambda + \mu} \tag{16}$$

The steady-state probability of the aggregate model is equivalent to that of the original model.

Proof. Step 1: Consider the Kolmogorov differential equation in steady-state:

$$\pi \times Q = 0 \tag{17}$$

π is the steady-state probability vector, Q is the transfer rate matrix, we have:

$$\pi_i \cdot q_i = \sum_{j \neq i} \pi_j \cdot q_{j,i} \tag{18}$$

$$\pi_i (\sum_{k \neq i} q_{i,k}) = \sum_{j \neq i} \pi_j \cdot q_{j,i} \tag{19}$$

Step 2: Without loss of generality, for analysis and proof, the up-states are numbered from 1 to h and the set of down-states from $h+1$ to n;

$$\sum_{i=1}^{h} \pi_i (\sum_{k \neq i} q_{i,k}) = \sum_{h+1}^{n} \sum_{j \neq i} \pi_j \cdot q_{j,i} \tag{20}$$

$$\begin{aligned} &\sum_{i=1}^{h} \pi_i \cdot (\sum_{k=1, k \neq i}^{h} q_{i,k} + \sum_{k=h+1}^{n} q_{i,k}) \\ &= \sum_{i=1}^{h} (\sum_{j=1, j \neq i}^{h} \pi_j q_{j,i} + \sum_{j=h+1}^{n} \pi_j q_{j,i}) \end{aligned} \tag{21}$$

then:

$$\begin{aligned} &\sum_{i=1}^{h} \pi_i \cdot \sum_{k=1, k \neq i}^{h} q_{i,k} + \sum_{i=1}^{h} \pi_i \cdot \sum_{k=h+1}^{n} q_{i,k} \\ &= \sum_{i=1}^{h} \sum_{j=1, j \neq i}^{h} \pi_j \cdot q_{i,j} + \sum_{i=1}^{h} \sum_{j=h+1}^{n} \pi_j \cdot q_{j,i} \end{aligned} \tag{22}$$

To make the first item on the left and the first item on the right equal, the order of summation can be changed, thus

$$\sum_{i=1}^{h} \sum_{j=h+1}^{n} \pi_j \cdot q_{j,i} = \sum_{i=1}^{h} \pi_i \cdot \sum_{k=h+1}^{n} q_{i,k} \tag{23}$$

then:

$$\sum_{t_{i,j} \subseteq RT} \pi_i \cdot q_{i,j} = \sum_{t_{i,j} \subseteq GT} \pi_i \cdot q_{i,j} \tag{24}$$

Therefore, we have

$$\frac{\lambda}{\mu} = \frac{\frac{1}{\pi_R} (\sum_{t_{i,j} \subseteq RT} \pi_i \cdot q_{i,j})}{\frac{1}{\pi_F} (\sum_{t_{i,j} \subseteq GT} \pi_i \cdot q_{i,j})} = \frac{1/\pi_R}{1/\pi_F} = \frac{\pi_F}{\pi_R} \tag{25}$$

$$\lambda = \frac{\pi_F}{\pi_R} \cdot \mu \tag{26}$$

From (11)–(13), and $\pi_R + \pi_F = 1$, it is shown

$$\pi_{R_{ag}} = \frac{\mu}{\frac{\pi_F}{\pi_R} \cdot \mu + \mu} = \frac{1}{\frac{\pi_F}{\pi_R} + 1} = \frac{\pi_R}{\pi_R + \pi_F} = \pi_R \tag{27}$$

The proof for showing $\pi_{F_{ag}} = \pi_F$ is similar.

4.2 Dependability Model of Edge Computing Systems

Edge computing applications use the computational, sensor, and networking resources of nearby mobile and stationary computing devices. Therefore, in contrast to cloud computing, edge computing processes data locally near to its origin [30], there by reducing the network transmission load and communication latency. Edge computing and cloud computing cooperate, and the proposed model is shown in Fig. 3. This article assumes that as long as either the edge or

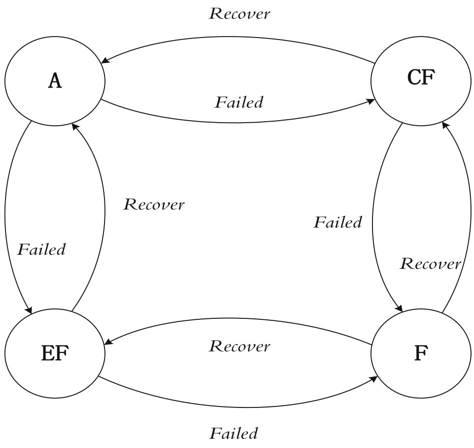

Fig. 3. Dependability model of edge-cloud system

the cloud is in a normal state, the system is reliable at this time. If both the edge and the cloud fail, the system fails. According to the hypothesis, we can draw four possible states of the service system: The state A means that both the edge server and the cloud server can process user requests normally; State EF means that the cloud server can process user requests normally, and the edge server is faulty, and the system is reliable; State CF means that the cloud server is faulty, but the edge server is operating normally and the system is reliable; The state F represents that both the edge server and the cloud server are faulty, and the system fails at this time.

In order to evaluate the dependability attributes, a state diagram based on a continuous-time Markov chain (CTMC) is established. The CTMC transition rate matrix is represented by Q, as shown below:

$$Q = \begin{bmatrix} q_{A,A} & q_{CF} & q_{EF} & 0 \\ q_A & q_{CF,CF} & 0 & q_F \\ q_A & 0 & q_{EF,EF} & q_F \\ 0 & q_{CF} & q_{EF} & q_{F,F} \end{bmatrix},$$

where,

$$q_{i,i} = -\sum_{j \neq i} q_{i,j}, i \subseteq A, CF, EF, F \tag{28}$$

The steady-state probability in each state, saying $\pi = [\pi_A, \pi_{CF}, \pi_{EF}, \pi_F]$, can be obtained using the following expressions:

$$\begin{cases} \pi \quad \cdot Q = 0 \\ \sum_i \pi_i = 1 \end{cases} \tag{29}$$

5 Model and Approach of Dependability Optimization

In the design and management of the service computing system, reliability is one of the important factors that cannot be ignored. In a real service computing system, the system is continuous in service, rather than discrete. When tasks are offloaded to the server, it is guaranteed that the system can effectively process user requests. This issue is simulated as a Continuous-time Markov decision process (CTMDP) problem through the state transition model [31]. A five tuple can be used to describe the CTMDP problem typically: $\{S, S_0, A(i), q^\mu(j \mid i), r^\mu(i)\}$, where the state space S is a finite set of fully observable states of the system. S_0 is the initial state, and $A(i)$ a family of measurable subsets of actions applicable in $i \subseteq S$. The policy μ is the set of all actions selected in every state, i.e., $\mu = [a_1, a_2 ... a_{|S|}]$ and $\mu(i) = a_i \subseteq A(i)$, $q^\mu(j|i)$ is an infinitesimal generator of transition probability following a specific formula, the formula is as follows:

$$P^\mu(t) = e^{tq^\mu(j|i)} \tag{30}$$

In order to maximize the expected average reward function $\bar{J}^\mu$ under the policy μ, it is necessary to find the optimal policy $\mu = \mu^*$ and define it as the sum of future rewards within an infinite horizon.

$$\overline{J}^\mu(i) = \lim_{T \to \infty} \frac{1}{T} E\{\int_0^T r^\mu(i)(X_t) dt | X_0 = i\} \tag{31}$$

The $\overline{J}^\mu$ converges to a value denoted as η^μ for any initial state i is

$$\eta^\mu = \overline{J}^\mu(i) = \pi^\mu r \tag{32}$$

Algorithm 1 :The Policy Iteration Algorithm

1: Let the initial policy to μ_0, and $k = 0$.
2: Using (32) and (35), get the performance potential function g^{μ_k}.
3: Improve the policy by:

$$\mu_{k+1} = argmax\{r^{\mu} + Q^{\mu} g^{\mu_k}\}$$

4: if $\mu_{k+1} = \mu_k$ then
5: Set $\mu^* = \mu_{k+1}$.
6: Stop.
7: else
8: Set $k = k + 1$.
9: Go to Step 2.
10: end if.

where π^{μ} is the steady state probability, and η^{μ} is shown as

$$\eta_k = \frac{1}{T_k} \int_0^{T_k} r(X_t) dt \tag{33}$$

The relationship between steady-state probability and the transfer rate matrix is shown in (29). Markov processes can be divided into embedded Markov chains and the sojourn time of each state, the transition probabilities of the embedded Markov chain P^{μ} satisfies the equation

$$Q_{\mu} = \Lambda^{\mu}(P^{\mu} - I) \tag{34}$$

where $\Lambda^{\mu} = diag[\lambda_1^{\mu}, \lambda_2^{\mu}, ...]$, $\lambda_i^{\mu} > 0$ represents the transfer rate of state i. The length of time represented by the state i can satisfy the exponential probability distribution. The CTMDP model will support Poisson equation:

$$Q^{\mu} g^{\mu} = -r^{\mu} + \eta^{\mu} e \tag{35}$$

where $g^{\mu}(i)$ is expressed as the performance potential function of state i. The performance potential function $g^{\mu}(i)$ is the difference between the instantaneous return and the long-term average return, the expectation in the infinite time domain. For different state paths, the long-term average return of the system is only related to the initial state. If a state is selected as the initial state and its performance potential is defined as $g(i) = 0$, then the long-term of other states and states i is the difference between average returns in performance potential. In addition, for the CTMDP model, the performance potential function can be expressed by the following formula:

$$g^{\mu}(i) = \lim_{T \to \infty} \frac{1}{T} E\{\int_0^T \{r^{\mu}(X_t) - \eta^{\mu}\} dt | X_0 = i\} \tag{36}$$

In this paper, we only consider the case of policy iteration algorithms. The policy algorithm used to solve the CTMDP problem will be introduced in this section. Firstly, the reward difference formula between the two strategies, namely μ_1 and μ_2, we have

$$\begin{aligned} &\eta^{\mu_1} - \eta^{\mu_2} \\ &= \pi^{\mu_1} r^{\mu_1} - \pi^{\mu_2} e \eta^{\mu_2} \\ &= \pi^{\mu_1} [(r^{\mu_1} + Q^{\mu_1} g^{\mu_2}) - (r^{\mu_2} + Q^{\mu_2} g^{\mu_2})] \end{aligned} \tag{37}$$

Definition 1. *If there is at least one vector with the same dimension, which can satisfy $u(i) \preceq v(i)$, and the rest are equal, it can be expressed as $u \preceq v$. Similarly, $u \succeq v$ can be defined in the same way as above.*

Theorem 1. *μ_1 and μ_2 are two policies of the CTMDP problem, if $r^{\mu_1} + Q^{\mu_1} g^{\mu_2} \succeq r^{\mu_2} + Q^{\mu_2} g^{\mu_2}$, then $\eta^{\mu_1} \succeq \eta^{\mu_2}$; Moreover, if $r^{\mu_1} + Q^{\mu_1} g^{\mu_2} \preceq r^{\mu_2} + Q^{\mu_2} g^{\mu_2}$, then $\eta^{\mu_1} \preceq \eta^{\mu_2}$.*

Proof. The theorem can be easily obtained based on Definition 1, (37) and the fact that the steady-state probability of an ergodic Markov process is greater than 0.

Theorem 2. *Consider a CTMDP problem. For any $\mu \subseteq D$, the strategy μ^* can be called the optimal strategy.*

$$r^{\mu^*} + Q^{\mu^*} g^{\mu^*} \succeq r^{\mu} + Q^{\mu} g^{\mu} \tag{38}$$

Proof. We first need to prove the necessary conditions. Assume that μ^* is the optimal strategy. We need to prove the validity of (38). Suppose there is a policy μ^{ξ} that makes (38) invalid, that is, there is at least one state i that can satisfy the following

$$r^{\mu^*}(i) + \sum_{j=1}^{S} q^{\mu^*}(j|i) g^{\mu^*}(j) < r^{\mu^{\xi}}(i) + \sum_{j=1}^{S} q^{\mu^{\xi}}(j|i) g^{\mu^*}(j)$$

Construct a new policy $\hat{\mu}$ in this way $\hat{\mu}(j) = \mu^*(j)$ while $j \neq i$; $\hat{\mu}(j) = \mu^*(j)$ while $j = i$. Thus, we can get $r^{\mu^*} + Q^{\mu^*} g^{\mu^*} \succeq r^{\mu} + Q^{\mu} g^{\mu}$. By applying Theorem 2, we see that $\eta^{\hat{\mu}} > \eta^{\mu^*}$ which contradicts to the previous assumption.

Next, we will give sufficient conditions to carry out a proof. Assume that (38) holds. Based on the application of Theorem 2, we can find that for any D, $\eta^{\mu^*} > \eta^{\mu}$, therefore, μ^* is the optimal strategy.

The previous description proves that as the theoretical basis, the strategy iteration algorithm is as Algorithm 1. In practical problems, due to the uncertainty or large scale of the state space, it is almost impossible to solve the Poisson equation. The structure of the evaluation strategy is evaluated through multiple sample paths and return values using the temporal differences (TD) method, which is a very effective method [32]. This section will introduce the TD algorithm used to evaluate the performance potential function.

Let $\{T_0, T_1, ...\}$ be a transition time sequence in CTMDP, when $t \subseteq [T_k, T_{k+1}]$, $x_t = X_k, k = \{0, 1, 2, 3...\}$. The sojourn time of state X_k is expressed

Algorithm 2 :g^{μ} evaluation algorithm

1: Let $k = 0$, $\eta^{\mu} = 0$, $g^{\mu}(i) = 0$, $I_i = 0$, where $I(i)$ represents the occurrence time of state i, $i \subseteq S$
2: Select a small constant $\xi > 0$
3: $k \leftarrow k + 1$
4: $I_{X_k} \leftarrow I_{X_k+1}$.
5: $\eta_{\mu} \leftarrow \eta_{\mu} + \frac{T_{k+1}-T_k}{T_{k+1}}[r^{\mu}(X_k) - \eta^{\mu}]$
6: $g^{\mu} \leftarrow g^{\mu} + s_k\{[r^{\mu}(X_t) - \eta^{\mu}]\tau_k\} + g^{\mu}(X_{k+1} - g^{\mu}(X_k)\})$
7: if $|g_{k+1} - g_k|_{max} < \xi$, then
8: output: g^{μ}
9: STOP.
10: else
11: Go to Step 2.
12: end if

as $\tau(X_k) = T_{k+1} - T_k$, use the (37) to get the approximate solution of g^{μ}, where s_k indicates the number of iterations:

$$g^{\mu} \approx g^{\mu} + s_k\{[r^{\mu}(X_t) - \eta^{\mu}]\tau_k\} + g^{\mu}(X_{k+1} - g^{\mu}(X_k)\}) \tag{39}$$

Using (32), we get

$$\begin{aligned}
\eta^{\mu}_{k+1} &= \frac{1}{T_{k+1}} \int_0^{T_{k+1}} r^{\mu}(X_t)dt \\
&= \frac{1}{T_{k+1}}[\int_0^{T_k} r^{\mu}(X_t)dt + r^{\mu}(X_t)(T_{k+1} - T_k)] \\
&= \frac{T_k}{T_{k+1}}\eta^{\mu}_k + \frac{T_{k+1} - T_k}{T_{k+1}} r^{\mu}(X_k) \\
&= \eta^{\mu}_k + \frac{T_{k+1} - T_k}{T_{k+1}}[r^{\mu}(X_k) - \eta^{\mu}_k]
\end{aligned} \tag{40}$$

Then, the average average reward value η_{μ} can be approximately expressed as

$$\eta_{\mu} \approx \eta_{\mu} + \frac{T_{k+1} - T_k}{T_{k+1}}[r^{\mu}(X_k) - \eta^{\mu}] \tag{41}$$

Use the (39) and (41) to get the Algorithm 2 for evaluating the performance potential function.

6 Empirical Evaluation

6.1 Data Set and Experimental Settings

In this article, a large high-performance service system is used, which is provided by $LANL$ [8]. This data set contains the failure and repair times of more than 20

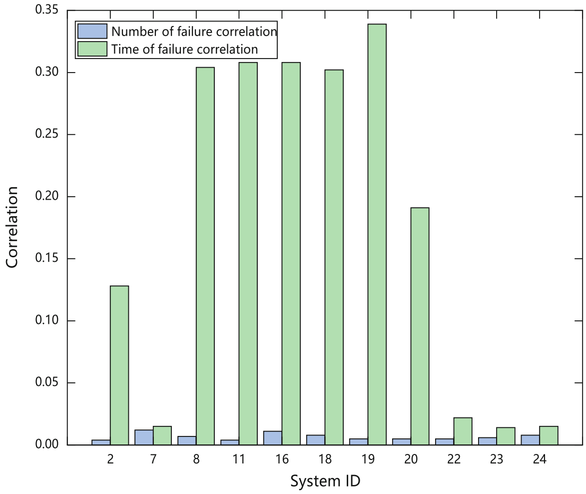

Fig. 4. The correlation analysis of failures in the system

different systems collected from 1996 to 2005. When the system fault is detected, the operator will establish a fault record, and the system administrator will notify the operator to fill in the end time of the fault after the successful repair. Since the number of nodes and processors in some systems differs by hundreds or thousands of times, they share the same characteristics with the ability of different edge computing and cloud computing to provide services in reality. Therefore, this data set is used to confirm our method.

When analyzing computer system failures, it is necessary to consider the correlation between the failures, and use the large-scale service computing system the $LANL$ data set for analysis. Through the analysis, it can be concluded that some systems have failure correlations, as shown in Fig. 4. Although there is not much difference in the proportion of the number of fault correlation to the total number of faults, there is a big difference in the time of fault correlation. Therefore, the consideration of fault correlation cannot be ignored. There are six types of failures in the data set, which are software, human error, network, facilities, hardware and undetermined, facility failures are caused by the environments. By analyzing the correlation between failures, the dependability of the system can be further accurately evaluated. Through the analysis of typical failures in the data set, as shown in the Fig. 5, the human failures destroy the integrity of the system, and the network failures cause the system to change from accessible state to inaccessible state. Under certain circumstances, this part of the failures can be accepted to avoid system failure. In addition, the analysis considers software, facilities, hardware, and make the system enter to the failed, and they are considered to be repairable in this paper. As users and suppliers define the level of catastrophe, in the case, we believe the undetermined failures are regarded as catastrophic consequences.

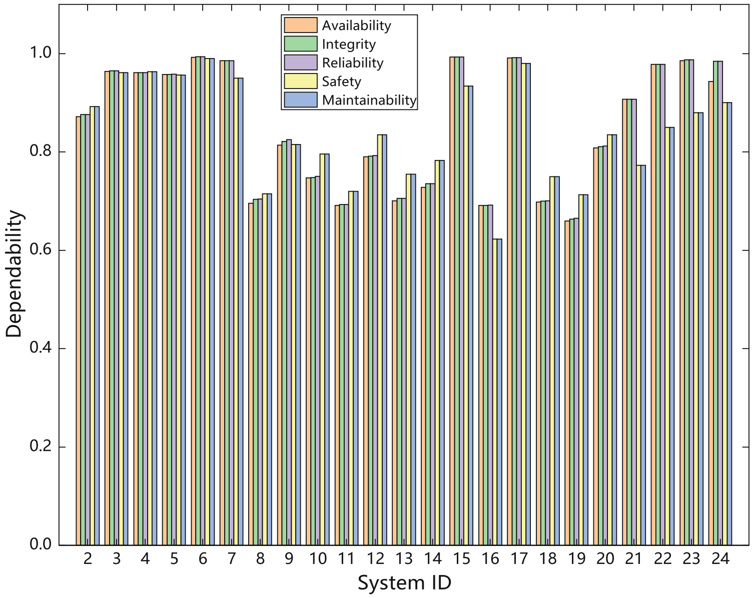

Fig. 5. Dependability attributes in the system

6.2 Experimental Results

The theorem is used to prove the correctness of the Algorithm 1, since in the real system the state space is usually large, therefore, it is almost impossible to solve the Poisson equation, Algorithm 2 is used to evaluate the performance potential function. This article uses the proposed edge and cloud collaboration model to confirm Algorithm 2. The return value for the fault state is set to 0, and the return value for the normal state is set to 1, the stay time of the system satisfies the exponential distribution, we have repeated the simulation 10 times, and each state transition is 10,000 times as the path, and the average is taken. The dynamic process of the performance potential function is shown in Fig. 6.

We first regularize the obtained approximate g^{μ} value and then calculate the mean and variance of the deviation from the theoretical value, the outcome are presented in Table 1. In order to further evaluate the performance of Algorithm 2, the L-step algorithm of CTMDP performance potentials used for comparison with algorithm. Since the L-step algorithm cannot obtain a graph of the change in performance potential with the number of transitions, the two are compared. The result obtained by the algorithm and the mean and variance of the theoretical deviation are used to evaluate the performance of our algorithm. The results of the L-step algorithm are shown in Table 2. The experimental results show

Table 1. Means and variances of the L-step evaluating results

States	g^{μ}	Means	Variances
A	2.9407	2.8509	9.3251×10^{-3}
CF	2.7870	2.6211	7.1072×10^{-3}
EF	2.4443	1.7946	1.0102×10^{-2}
F	−3.8787	−3.6148	1.2947×10^{-2}

Table 2. Means and variances of the TD evaluating results

States	g^{μ}	Means	Variances
A	2.9407	0.0718	6.6579×10^{-3}
CF	2.7870	0.0675	6.4457×10^{-3}
EF	2.4443	0.0676	9.1805×10^{-3}
F	−3.8787	−0.0959	1.2130×10^{-2}

that the TD algorithm is better than the L-step algorithm in terms of mean and variance. The deviation of the theoretical value is smaller than that of the L-step method.

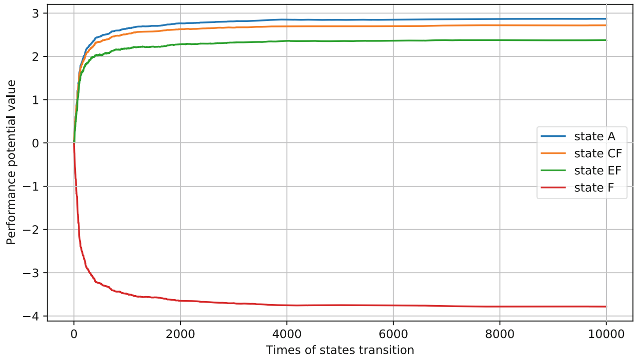

Fig. 6. Simulation state process

7 Conclusion

With the increasing demand on low latency and high privacy, computing models that combine edge computing and traditional cloud computing have been widely used, and therefore, dependability has become one of the key challenges. In this paper, we establish a dependability model, which takes into account the failure correlation, embeds the discrete-time Markov chain, and quantitatively analyzes the attributes of dependability. For a cloud-edge computing system, we propose a state aggregation approach, and prove the equivalence of the models before and after aggregation. Moreover, we formulate the dependability optimization problem as a continuous-time Markov decision process, and present effective algorithms to solve the problem. Finally, we evaluate our approach by simulation experiments based on real-world dataset. In future, we will continue to analyze the large-scale open-source data, propose new models capturing the dynamics of edge computing systems, and design new methods with higher efficiency.

Acknowledgment. This work is supported by Beijing Nova Program (No. Z201100006820082), National Natural Science Foundation of China (No. 61972414), National Key Research and Development Plan (No. 2016YFC0303700), Beijing Natural Science Foundation (No. 4202066), and the Fundamental Research Funds for Central Universities (Nos. 2462018YJRC040 and 2462020YJRC001).

References

1. Ozcan, M.O., Odaci, F., Ari, I.: Remote debugging for containerized applications in edge computing environments. In: 2019 IEEE International Conference on Edge Computing (EDGE), pp. 30–32 (2019). https://doi.org/10.1109/EDGE.2019.00021
2. Amanatullah, Y., Lim, C., Ipung, H.P., Juliandri, A.: Toward cloud computing reference architecture: cloud service management perspective. In: International Conference on ICT for Smart Society, pp. 1–4 (2013). https://doi.org/10.1109/ICTSS.2013.6588059
3. Wei, X., et al.: MVR: an architecture for computation offloading in mobile edge computing. In: 2017 IEEE International Conference on Edge Computing (EDGE), pp. 232–235 (2017). https://doi.org/10.1109/IEEE.EDGE.2017.42
4. Xu, J., Palanisamy, B., Ludwig, H., Wang, Q.: Zenith: utility-aware resource allocation for edge computing. In: 2017 IEEE International Conference on Edge Computing (EDGE), pp. 47–54 (2017). https://doi.org/10.1109/IEEE.EDGE.2017.15
5. Loghin, D., Ramapantulu, L., Teo, Y.M.: Towards analyzing the performance of hybrid edge-cloud processing. In: 2019 IEEE International Conference on Edge Computing (EDGE), pp. 87–94 (2019). https://doi.org/10.1109/EDGE.2019.00029
6. Jain, R., Tata, S.: Cloud to edge: distributed deployment of process-aware IoT applications. In: 2017 IEEE International Conference on Edge Computing (EDGE), pp. 182–189 (2017). https://doi.org/10.1109/IEEE.EDGE.2017.32
7. Esteves-Verissimo, P., Völp, M., Decouchant, J., Rahli, V., Rocha, F.: Meeting the challenges of critical and extreme dependability and security. In: 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC), pp. 92–97 (2017). https://doi.org/10.1109/PRDC.2017.21
8. Schroeder, B., Gibson, G.A.: A large-scale study of failures in high-performance computing systems. IEEE Trans. Dependable Secure Comput. **7**(4), 337–350 (2010). https://doi.org/10.1109/TDSC.2009.4
9. Pan, Y., Hu, N.: Research on dependability of cloud computing systems. In: 2014 10th International Conference on Reliability, Maintainability and Safety (ICRMS), pp. 435–439 (2014). https://doi.org/10.1109/ICRMS.2014.7107234

10. Guan, Q., Chiu, C., Fu, S.: CDA: a cloud dependability analysis framework for characterizing system dependability in cloud computing infrastructures. In: 2012 IEEE 18th Pacific Rim International Symposium on Dependable Computing, pp. 11–20 (2012). https://doi.org/10.1109/PRDC.2012.10
11. Walunj, S.G., Nagrare, T.H.: Dependability issues on cloud environment and analyzing server responsibilities. In: 2018 2nd International Conference on Inventive Systems and Control (ICISC), pp. 926–928 (2018). https://doi.org/10.1109/ICISC.2018.8398936
12. Qiu, X., Luo, L., Dai, Y.: Reliability-performance-energy joint modeling and optimization for a big data task. In: 2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 334–338 (2016). https://doi.org/10.1109/QRS-C.2016.51
13. Mondal, S.K., Sabyasachi, A.S., Muppala, J.K.: On dependability, cost and security trade-off in cloud data centers. In: 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC), pp. 11–19 (2017). https://doi.org/10.1109/PRDC.2017.12
14. Bai, Y., Zhang, H., Fu, Y.: Reliability modeling and analysis of cloud service based on complex network. In: 2016 Prognostics and System Health Management Conference (PHM-Chengdu), pp. 1–5 (2016). https://doi.org/10.1109/PHM.2016.7819907
15. Zhang, N., Li, R.: Resource optimization with reliability consideration in cloud computing. In: 2016 Annual Reliability and Maintainability Symposium (RAMS), pp. 1–6 (2016). https://doi.org/10.1109/RAMS.2016.7447982
16. Luo, L., Li, H., Qiu, X., Tang, Y.: A resource optimization algorithm of cloud data center based on correlated model of reliability, performance and energy. In: 2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 416–417 (2016). https://doi.org/10.1109/QRS-C.2016.69
17. Chowdhury, A., Tripathi, P.: Enhancing cloud computing reliability using efficient scheduling by providing reliability as a service. In: 2014 International Conference on Parallel, Distributed and Grid Computing, pp. 99–104 (2014). https://doi.org/10.1109/PDGC.2014.7030723
18. Dastjerdi, A.V., Buyya, R.: An autonomous reliability-aware negotiation strategy for cloud computing environments. In: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012), pp. 284–291 (2012). https://doi.org/10.1109/CCGrid.2012.101
19. Song, Y., Yau, S.S., Yu, R., Zhang, X., Xue, G.: An approach to QoS-based task distribution in edge computing networks for IoT applications. In: 2017 IEEE International Conference on Edge Computing (EDGE), pp. 32–39 (2017). https://doi.org/10.1109/IEEE.EDGE.2017.50
20. Caprolu, M., Di Pietro, R., Lombardi, F., Raponi, S.: Edge computing perspectives: architectures, technologies, and open security issues. In: 2019 IEEE International Conference on Edge Computing (EDGE), pp. 116–123 (2019). https://doi.org/10.1109/EDGE.2019.00035
21. Martín Fernãndez, C., Díaz Rodríguez, M., Rubio Muñoz, B.: An edge computing architecture in the internet of things. In: 2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC), pp. 99–102 (2018). https://doi.org/10.1109/ISORC.2018.00021
22. Chen, X., Liu, W., Chen, J., Zhou, J.: An edge server placement algorithm in edge computing environment. In: 2020 12th International Conference on Advanced Infocomm Technology (ICAIT), pp. 85–89 (2020). https://doi.org/10.1109/ICAIT51223.2020.9315526

23. Badri, H., Bahreini, T., Grosu, D., Yang, K.: Risk-based optimization of resource provisioning in mobile edge computing. In: 2018 IEEE/ACM Symposium on Edge Computing (SEC), pp. 328–330 (2018). https://doi.org/10.1109/SEC.2018.00033
24. Xiao, K., Gao, Z., Wang, Q., Yang, Y.: A heuristic algorithm based on resource requirements forecasting for server placement in edge computing. In: 2018 IEEE/ACM Symposium on Edge Computing (SEC), pp. 354–355 (2018). https://doi.org/10.1109/SEC.2018.00043
25. Ribeiro, R., Favarim, F., Barbosa, M.A.C., Koerich, A.L., Enembreck, F.: Combining learning algorithms: an approach to Markov decision processes. In: Cordeiro, J., Maciaszek, L.A., Filipe, J. (eds.) ICEIS 2012. LNBIP, vol. 141, pp. 172–188. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40654-6_11
26. Huang, J., Lin, C., Kong, X., Wei, B., Shen, X.: Modeling and analysis of dependability attributes for services computing systems. IEEE Trans. Serv. Comput. **7**(4), 599–613 (2014). https://doi.org/10.1109/TSC.2013.8
27. Avizienis, A., Laprie, J., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput. **1**(1), 11–33 (2004). https://doi.org/10.1109/TDSC.2004.2
28. Lanus, M., Yin, L., Trivedi, K.S.: Hierarchical composition and aggregation of state-based availability and performability models. IEEE Trans. Reliab. **52**, 44–52 (2003)
29. Stewart, W.J.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton (1994)
30. Zheng, S., Tilevich, E.: A programming model for reliable and efficient edge-based execution under resource variability. In: 2019 IEEE International Conference on Edge Computing (EDGE) (2019)
31. Huang, J., Lin, C., Wan, J.: Modeling, analysis and optimization of dependability-aware energy efficiency in services computing systems. In: 2013 IEEE International Conference on Services Computing, pp. 683–690. IEEE (2013)
32. Jia, S., Shen, L., Xue, H.: Continuous-time Markov decision process with average reward: using reinforcement learning method. In: 2015 34th Chinese Control Conference (CCC), pp. 3097–3100 (2015). https://doi.org/10.1109/ChiCC.2015.7260117