Received July 4, 2021, accepted August 4, 2021, date of publication August 18, 2021, date of current version August 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3105910

Performability Analysis of a Redundant Parallel Task in Network Systems

MIN TAO[1], XIWEI QIU[2], AND PENG SUN[2]
[1]School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China
[2]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China

Corresponding author: Xiwei Qiu (qiuxw@uestc.edu.cn)

This work was supported in part by the Sub-Project of the Independent Scientific Research Project through the University of Electronic Science and Technology of China under Grant ZZKY-ZX-03-02-04.

ABSTRACT Recently, application of network systems (*e.g.*, cloud computing systems) is increasingly prevalent for achieving integration, sharing and efficient utilization of various resources. A parallel task in a network system has multiple subtasks that can be executed in different servers in parallel. However, failures of any subtask inevitably result in that the entire task cannot be complete. To avoid such a situation, the network system can create some copies from a subtask and make them run on different servers simultaneously. This redundant parallel execution manner is an efficient approach to improve performance and guarantee reliability. However, it also brings complexity in modeling, evaluation and optimization. For example, link failures inevitably lead to inaccessibility of some servers, and server failures also result in that subtasks hosted on the server cannot be complete. This is the complicated failure correlation that cannot be ignored in modeling and evaluation. This paper first presents a reliability-performance correlation model for a redundant parallel task in the network system. The model captures precedence constraints of subtasks, multiple types of failures and complicated failure correlations to improve fidelity. This paper also design an algorithm that encompasses the Graph theory and the Bayesian theorem to evaluate a performability metric, which can be used to quantify important reliability-performance correlation. Finally, a heuristic algorithm is designed to search an optimal task execution strategy that maximizes the performability metric. Illustrative examples are presented.

INDEX TERMS Network systems, precedence constraints, failure correlations, redundant parallel computing.

I. INTRODUCTION

Network systems have became increasingly important for achieving efficient use of various computing resources via internet. Recently, the most important application of the network system is the cloud computing system, especially, the infrastructure-as-a-service (IaaS) system [1]. Although the network system can integrate and share heterogeneous resources, it also brings an important kind of failures that cannot be ignored, *i.e.*, network failures.

In real-world application scenarios of the network system, many tasks have high computational complexities, *e.g.*, big data processing tasks. For efficiently reducing the completion time of such a task, the task can be divided into multiple subtasks to be executed in parallel. However, this parallel computing manner also brings a negative effect on task completion. That is, if any subtask is failed, the entire task cannot be complete. Therefore, replication is usually adopted to achieve fault tolerant.

In many traditional systems, redundant computing is only triggered after a subtask is found to be fail. However, it is not an efficient approach from the perspective of performance, especially for some subtasks that have precedent constraints. Therefore, to guarantee reliability and performance simultaneously, a more efficient redundant computing manner is that each subtask has multiple redundant copies running simultaneously [2].

In principle, the network system has a centralized control node (CN) that makes a decision on how to execute a parallel task rationally. Not only performance but also reliability should be considered together. The performability metric can be used to quantify random performance affected by random

The associate editor coordinating the review of this manuscript and approving it for publication was Zhaojun Li.

reliability factors [3]. However, since the network system usually has a complexity network structure, modeling, analyzing and evaluating the performability of a redundant parallel task in such a system is indeed difficult.

In fact, there are multiple types of failures in the network system, including task failures, server failures, and link failures [4]. More importantly, some complex failure correlations exist among those failures. For example, failures of a link inevitably result in that servers connecting to the CN through the link become inaccessible, and failures of a server also lead to subtasks hosted on the server are failed. Meanwhile, precedence constraints of subtasks also bring failure correlations among those subtasks. For instance, a data mining task may have multiple subtasks of data checking, data cleaning, data processing, and result verification that are executed in sequence. Failures of a predecessor subtask definitely make that the successor subtasks cannot be complete. Therefore, performance evaluation should capture random changes caused by various kinds of failures and the corresponding failure correlations for ensuring necessary precision.

The notion of performability is proposed for combining reliability with performance [5]. Performability models for traditional systems (*e.g.*, fault-tolerant computer systems [6]) cannot be directly extended to the network system as the complicated network structure is not taken into account. Many recent researches focused on performability analysis of cloud computing systems. Some important performability metrics, such as expected response delay, expected waiting time, and expected execution time of cloud services were evaluated based on different stochastic models in reliability [7], [8]. However, those researches do not consider important failure correlations. Although our prior research [9] investigated failure correlation between co-located virtual machines and the host server in the proposed performability model, it cannot be directly applied to a scenario that a task is executed in the network system since the parallel and redundant computing is not taken into account.

Another important research filed related to the network system is how to design an optimal resource scheduling strategy. Considerable researches have studied many optimization techniques for satisfying various optimization objectives, such as using a data aggregation technique to save energy consumption of a cloud computing platform [11], achieving load balancing of resources by adopting an optimal task scheduling technique [12], enhancing the security of a cloud system by using some optimal virtual machine allocation policies [13], and minimizing the cost of a cloud system by using a workflow scheduling approach [14]. However, those optimization techniques were usually proposed from the perspective of the system but not a task. Therefore, they cannot be directly adopted to find a fine-grained task execution strategy when reliability and performance are considered simultaneously.

There are many factors should be fully taken into account when the CN makes a rational task execution strategy. For example, how many redundant copies should be created for each subtask, and how to assign those copies to servers in the network system. In fact, excessive emphasis on redundancy may not be necessary and even becomes irrational sometimes. The increment of the reliability of a subtask gained by adding a redundant copy becomes increasingly slight with the increase of the number of its redundant copies. Therefore, for the network system where resources need to be shared by various tasks, excessive redundancy is not an efficient resource utilization pattern.

This paper systemically studies a theoretical performability model for a task executed in the parallel and redundant manner in the network system. The primary innovation of the model is that it captures precedence constraints among subtasks, multiple types of failures and the corresponding failure correlations existed in the network system. An algorithm based on the Bayesian theorem and the Graph theory is presented to obtain the probability distribution of random task completion time. The performability metric quantifying the reliability-performance correlation is further derived. Finally, this paper also develops a heuristics algorithm to search an optimal task execution strategy for maximizing the performability metric.

The remainder of the paper is organized as follows. Section 2 introduces an extended topological structure to describe the network system and analyze the existing failure correlations. Section 3 presents an algorithm encompassing the Graph theory and the Bayesian theorem to derive the performability metric of a redundant parallel task. Section 4 develops a heuristic algorithm to find an optimal task executing strategy maximizing the performability metric of the task. Numerical examples are illustrated in Section 5. Section 6 describes some related researches followed by highlights of new contributions made by this paper. Section 7 concludes this paper.

II. ANALYSIS OF THE NETWORK SYSTEM

A. DESCRIPTION OF THE NETWORK STRUCTURE

At present, the network system in a realistic environment is usually constructed by using cloud computing. The most important technique of the cloud computing is virtualization. This technique makes various heterogeneous resources can be shared by multiple tasks through an uniform access interface, *i.e.*, a control node (CN). The CN is a critical component of the current network system. It performs the centralized management of a large amount of heterogeneous resources.

When the CN receives a task with multiple subtasks, it becomes the start and the terminal points to execute the task. To improve the completion probability of the task, the CN first makes a decision on how many redundant copies should be created for each subtask (since each copy is essential a subtask, in the following content, we also use term 'subtask' mean a copy sometimes). Then, all the subtasks are assigned to a set of heterogeneous servers that are distributed in the network systems.

Some researches [15] suggested using a star topology to describe the system structure for simplifying analysis and

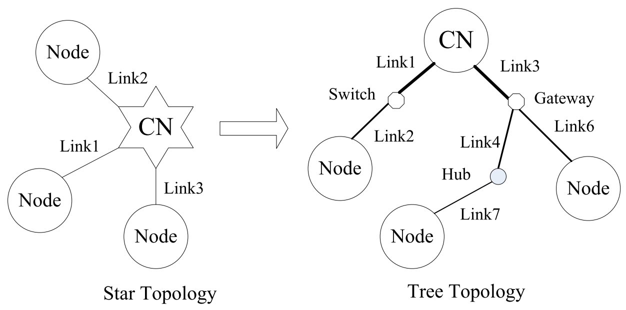

FIGURE 1. **Description of the network structure.**

computation, where the center is the CN, the node is servers, and each branch represents a link between the CN and a server. However, different from traditional computing systems, the network system is constructed based on the complicated internet environment. This makes that there are multiple different links between the CN and a server. If those different links are simplified into a branch, it is actually not an accurate approximation for the network system. Instead of using the star topology, we apply a more realistic tree topology to describe the network system structure, as shown in Fig. 1.

B. DEFINITION OF FAILURE CORRELATIONS

It is important to notice that reliability and performance are correlated and should not be treated separately [16]. For example, if any subtask becomes fail due to random failures, the entire task cannot be complete. Therefore, various types of failures and failure correlations should be considered comprehensively. Failures in the network system can be mainly summarized as three types, *i.e.*, link failures, server failures, and task failures. In general, failures of different servers, and failures of different links can be treated as independent of one another [4]. However, there still exist complicated failure correlations, which are described as follows.

1) *Server-Subtask (S-T) failure correlation*: failures of a server inevitably result in that subtasks hosted on it cannot operate. This can be treated as the failure correlation between a server and subtasks.
2) *Link-Server (L-S) failure correlation*: failures of a link also make that all servers connecting to the CN through it become inaccessible. That is, the failure correlation between a link and servers. Note that a L-S failure correlation may lead to multiple S-T failure correlations subsequently.
3) *Subtask-Subtask (T-T) failure correlation*: Another important factor bringing about the failure correlation is precedence constraints among subtasks (*e.g.*, a subtask must take the outputs of other subtasks as its own input). Apparently, if there does not exist any redundancy of subtasks, failures of a predecessor subtask definitely make that the successor subtasks cannot begin.

The model and algorithms presented in the following section capture both multiple types of failures and the

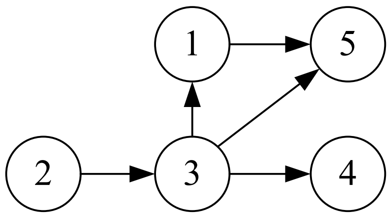

FIGURE 2. **An example of precedence constraints among subtasks.**

complicated failure correlations in performability evaluation for a redundant parallel task in the network system.

III. THE PERFORMABILITY MODEL

A. FAULT-FREE TASK COMPLETION TIME

Suppose a task submitted to the CN has a work requirement w, which can be quantified as the number of instructions or commands needed to be executed. The task is designed to have M ($M \geq 1$) subtasks in total. The work requirement of subtask m is w_m ($1 \leq m \leq M$), and equation $w = \sum_{m=1}^{M} w_m$ is satisfied. If subtask m is assigned to server n, the *fault-free execution time* (FFET) of subtask m is defined as the time only consumed in its execution without considering any failures, which can be obtained as

$$u_{mn} = \frac{w_m}{c_{mn}} \tag{1}$$

where c_{mn} is the computational speed of subtask m when it runs on server n. The value of c_{mn} depends on the resource capacity assigned to the subtask. There are many recent technologies can be used to perform the capability of assigning a specific computational resource capacity to a subtask. Typically, the cloud virtualization technology can not only guarantee non-interfering execution of co-located subtasks but also ensure each subtask exclusively occupies a CPU core [17]. In this paper, we also assume that a CPU core is exclusively occupied by a subtask. Suppose f_n is the maximal core frequency of multi-core processor deployed in server n. Then, c_{mn} can be written as $c_{mn} = f_n$.

We first analysis the situation that the CN does not make any redundant copies for the task. If there does not exist any precedence constraints among the subtasks, the subtasks can be executed totally in parallel. The entire task is complete only after the completion of all subtasks, and its *fault-free completion time* (FFCT) is defined as the time interval between its submission and its completion without considering any failures, which is written as

$$t = \max_{1 \leq m \leq M} (u_{mn}) \tag{2}$$

However, in realistic environments, precedence constraints among subtasks are usually existed, such as data dependencies, which make the subtasks cannot be executed totally in parallel. Fig. 2 shows an example of precedence constraints of subtasks. As shown in the figure, the task has five subtasks ($M = 5$). The completion of subtask 4 and 5 means that the entire task is successfully complete. The precedence

Algorithm 1 Calculate the FFCT of Subtask m

```
Require: $w_1 \dots w_M, f_1 \dots f_N, H$
Ensure: $t_m$
 1: function CFFCT(m)
 2:     $t_m \leftarrow 0$
 3:     $u_m \leftarrow w_m/f_n$ //FFET of subtask $m$
 4:     $S \leftarrow \text{sum}[H(m,:)]$ //the number of predecessor
        subtasks of subtask $m$
 5:     if $S = 0$ then //no predecessor subtask
 6:         $t_m \leftarrow u_m$
 7:     else if $S = 1$ then //one predecessor subtask
 8:         $j \leftarrow \text{find}[H(m,:) = 1]$
 9:         $t_m \leftarrow u_m + \text{CFFCT}(j)$
10:     else //multiple predecessor subtasks
11:         $[j_1, \dots, j_S] \leftarrow \text{find}[H(m,:) = 1]$
12:         $t_m \leftarrow u_m + \max[\text{CFFCT}(j_1), \dots, \text{CFFCT}(j_S)]$
13:     end if
14:     return $t_m$
15: end function
```

constraints of the subtasks can be expressed as a $M \times M$ matrix:

$$H = \begin{pmatrix} 0&0&1&0&0 \\ 0&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 1&0&1&0&0 \end{pmatrix} \tag{3}$$

Element h_{ij} in matrix H takes values of 0 or 1. If $h_{ij} = 1$, subtask i must begin after the completion of subtask j. The *fault-free waiting time* (FFWT) of subtask i is defined as the time from submitting it to a server to beginning it without considering any failures. If $h_{ij} = 0$, there does not exist a direct precedence constraint between subtask i and j. But this does not means that subtask i does not have a FFWT, as subtask i may have some other precedent subtasks. In general, the subtasks can be numbered arbitrarily. After assigning all the subtasks to specific servers, we can let v_m and u_m ($u_m = u_{mn}$) represent the FFWT and FFET of subtask m, respectively. The FFCT of subtask m can be calculated by $t_m = v_m + u_m$. The FFCT of subtask m can be calculated recursively, the corresponding pseudocode is shown in Algorithm 1. After obtaining the FFCTs of all the subtasks, the FFCT of the entire task can be calculated as

$$t = \max(t_1, \dots, t_m, \dots, t_M) \tag{4}$$

However, as mentioned in Section 2.2, failures of any subtask inevitably keep the entire task from completing. Thus, it is worth letting some servers execute replicates of the subtasks to improve reliability. Meanwhile, random failures also make the performance become a complex random variable. Therefore, the performability metric capturing the reliability-performance correlation is more precise than the FFCT metric for quantifying random performance of the task.

B. RANDOM TASK COMPLETION TIME

Fig. 3 shows a scenario of a redundant parallel task executed in the network system. As shown in the figure, the task have five subtasks ($M = 5$) with different work requirement w_m. From (1), the FFETs of the subtasks can be obtained, which have displayed in the figure.

1) ASSUMPTION

We first make the following assumptions for performability modeling:

1) Failures existed in the network system includes three types, *i.e.*, link failures, server failures, and subtask failures. Failure times of link k, server n and subtask m follow exponential distributions with failure rates of μ_k, θ_n and λ_m, respectively.
2) For achieving resource sharing, a server can host multiple subtasks. The network system have some technique (*e.g.*, cloud virtualization) to guarantee non-interfering execution of co-located subtasks that are hosted on an identical server.
3) A subtask can have multiple copies for improving its reliability. The system assigns a subtask and its copies to different servers to reduce the negative affect caused by server failures. As for a subtask executed in several servers redundantly, it is complete when the first server finishes it. The entire task is complete when all of the subtasks are complete.
4) When the CN makes a task execution strategy, it assigns all the subtasks (including their copies) to servers simultaneously. It also monitors the states of all subtasks real time. Once a predecessor subtask is complete, the CN can notify the corresponding successor subtasks immediately.
5) When a server hosts subtasks, not only itself but also the links connecting it with the CN must be available for completing the subtasks. This is because data, notification information and monitoring information must be translated through those links.
6) For a server in the waiting mode (*i.e.*, the subtasks hosted on it are waiting for the completion of their predecessor subtasks), it remains as hot-standby for avoiding time-consuming cold start.

Denote the link set connecting server n and the CN as φ_n. When subtask m is assigned to server n, its FFET u_{mn} can be obtained from (1). However, random failures make the realistic execution time become a random variable, denoted by U_{mn}. It can take two possible values:

$$U_{mn} = u_{mn} = \frac{w_m}{f_n} \tag{5}$$

if subtask m, server n and all links belonging to set φ_n do not fail until the subtask completion, and $U_{mn} = \infty$ otherwise. The probability that subtask m hosted on server n is successfully executed (denoted by π_{mn}) can be calculated as

$$\pi_{mn} = \exp\left[-(\lambda_m + \theta_n + \sum_{k \in \varphi_n} \mu_k) \cdot u_{mn}\right] \tag{6}$$

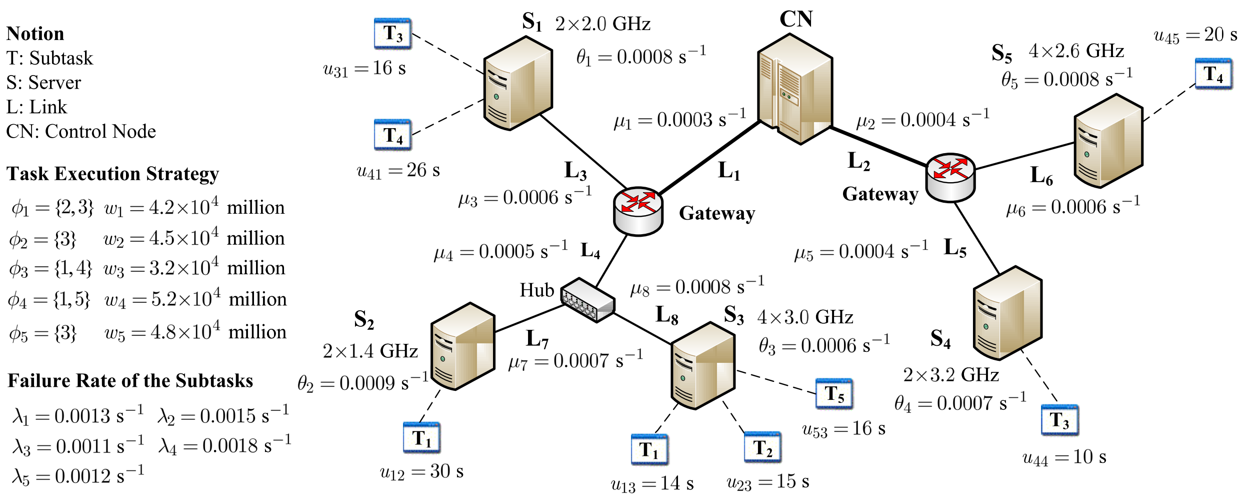
Notion
T: Subtask
S: Server
L: Link
CN: Control Node
Task Execution Strategy
$\phi_1=\{2,3\}$ $w_1=4.2\times10^4$ million
$\phi_2=\{3\}$ $w_2=4.5\times10^4$ million
$\phi_3=\{1,4\}$ $w_3=3.2\times10^4$ million
$\phi_4=\{1,5\}$ $w_4=5.2\times10^4$ million
$\phi_5=\{3\}$ $w_5=4.8\times10^4$ million
Failure Rate of the Subtasks
$\lambda_1=0.0013$ s^{-1} $\lambda_2=0.0015$ s^{-1}
$\lambda_3=0.0011$ s^{-1} $\lambda_4=0.0018$ s^{-1}
$\lambda_5=0.0012$ s^{-1}
S1 2×2.0 GHz $\theta_1=0.0008$ s^{-1}
CN
S5 4×2.6 GHz $\theta_5=0.0008$ s^{-1}
$u_{45}=20$ s
$u_{31}=16$ s
$u_{41}=26$ s
$\mu_1=0.0003$ s^{-1}
$\mu_2=0.0004$ s^{-1}
L3 $\mu_3=0.0006$ s^{-1}
L1
L2
Gateway
Gateway
L6 $\mu_6=0.0006$ s^{-1}
$\mu_4=0.0005$ s^{-1} L4
$\mu_5=0.0004$ s^{-1} L5
Hub
$\mu_8=0.0008$ s^{-1}
L8
S3 4×3.0 GHz $\theta_3=0.0006$ s^{-1}
S2 2×1.4 GHz $\theta_2=0.0009$ s^{-1}
L7 $\mu_7=0.0007$ s^{-1}
S4 2×3.2 GHz $\theta_4=0.0007$ s^{-1}
$u_{53}=16$ s
$u_{12}=30$ s
$u_{13}=14$ s
$u_{23}=15$ s
$u_{44}=10$ s
T1 T2 T3 T4 T5

FIGURE 3. A scenario of redundant parallel task executed in the network system.

These given the probability distribution of random execution time U_{mn}:

$$\begin{cases}\Pr(U_{mn}=u_{mn})=\pi_{mn}\\ \Pr(U_{mn}=\infty)=1-\pi_{mn}\end{cases}\tag{7}$$

According to assumption 3, subtask m may be redundantly executed in different servers. Since those heterogeneous servers have different computational capabilities (*e.g.*, maximal core frequencies), the FFETs of the redundant subtasks are also different. Suppose the servers executing subtask m compose a set ϕ_m ($\phi_m \neq \emptyset$). The random time for completing subtask m is the shortest time when one of the servers completes the subtask. Denote it as U_m, which can be written as

$$U_m = \min_{n\in\phi_m}(U_{mn})\tag{8}$$

Let T represent the random completion time of the entire task. If no precedence constraints exist among the subtasks, it can be written as

$$T = \max_{1\le m\le M}(U_m)\tag{9}$$

Although our prior research has proposed an algorithm to calculate T from (8) and (9) for the network system with the star topological structure [18], it did not consider precedence constraints among the subtasks and cannot be directly applied to the tree topological structure. To remedy this lack, this paper presents an algorithm based on the Graph theory and the Bayesian theorem to derive task completion time distribution for a more realistic scenario.

2) MINIMAL TASK EXECUTION TREE

A *Minimal Task Execution Tree* (MTET) is a minimal possible combination of necessary elements for guaranteeing the execution of the entire task. The elements in a MTET include subtasks, servers and links. Each MTET exactly contains M

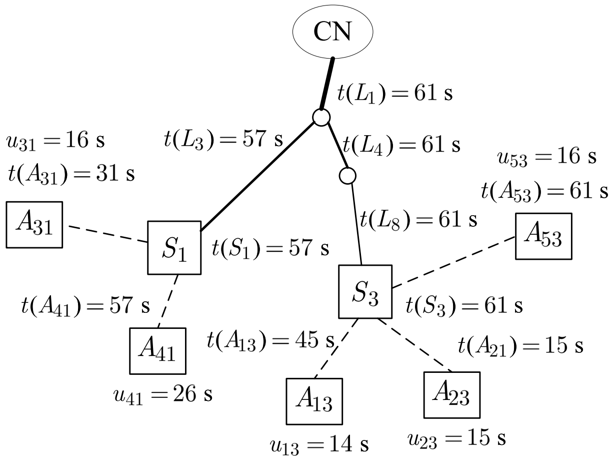

FIGURE 4. The spanning tree of a MTET.

different subtasks. It also contains the servers hosting those subtasks and the links connecting the servers with the CN.

All MTETs can be found by solving the corresponding graph traversal problem, such as the graph shown in Fig. 3. For an arbitrary graph, several classical algorithms such as Depth-First search and Breadth-First search can be applied to found all MTETs. Given a concrete redundancy strategy of all subtasks (*i.e.*, $\phi_1, \phi_2, \ldots, \phi_M$), the total number of all possible MTETs is

$$Z = \prod_{m=1}^{M}|\phi_m|\tag{10}$$

The element in a MTET can be expressed as a two-filed record - (*identifier*, *FFCT*). The identifier has three forms: 1) A_{mn} means that subtask m is assigned to server n; 2) S_n represents server n; 3) L_k is link k. Another field, *i.e.*, the FFCT can be interpreted as the minimal time that the corresponding element should keep available for guaranteeing the completion of the corresponding MTET.

For example, in Fig. 3, subtasks 1, 2 and 5 hosted in server 3 (A_{13}, A_{23}, A_{53}) and subtasks 3 and 4 hosted in server 1 (A_{31}, A_{41}) constitute a MTET. The corresponding spanning tree is shown in Fig. 4. Let $t(e_d)$ represent the FFCT of an element, where $e_d \in \{A_{mn}, S_n, L_k\}$ is the identifier of the element. First, the FFCTs of the subtasks can be derived from algorithm 1 presented in Section 3.1. Then, the FFCT of a server can be obtained as the maximum value of the FFCTs of the co-located subtasks hosted on the server, *e.g.*, $t(S_1) = \max(t(A_{31}), t(A_{41})) = 57$ s. Similarly, the FFCT of a link is calculated as the maximum value of the FFCTs of all servers that connect to the CN through it, such as $t(L_1) = \max(t(S_1), t(S_3)) = 61$ s. Finally, the MTET can be expressed as

$$\begin{aligned}\psi = \{&(A_{13}, 45), (A_{23}, 15), (A_{31}, 31), (A_{41}, 57), (A_{53}, 61)| \\ &(S_1, 57), (S_3, 61)|(L_1, 61), (L_4, 61), (L_8, 61), (L_3, 57)\}\end{aligned} \tag{11}$$

where ψ is the set of all element expressed as the two-field record. As seen in (11), we use '|' to separate different types of elements.

Suppose the zth MTET ($1 \leq z \leq Z$) has $D_z = |\psi_z|$ elements, and the dth element ($d = 1, 2, \ldots, D_z$) has failure rate ζ_d ($\zeta_d \in \{\lambda_m, \theta_n, \mu_k\}$). We can use F_z represent the event that the zth MTET finish the task, and its probability can be derived as

$$\Pr(F_z)=\Pr(\psi_z)=\prod_{d=1}^{D_z}\Pr(e_d)=\prod_{d=1}^{D_z}\exp\left(-\zeta_d \cdot t(e_d)\right) \tag{12}$$

where $\Pr(e_d)$ is the probability that element e_d does not fail during its FFCT $t(e_d)$, and $\Pr(\psi_z)$ is the probability that all elements in set ψ_z do not fail during their FFCTs.

3) EVALUATION ALGORITHM

In this section, we will present an evaluation algorithm to derive the probability distribution of random task completion time.

Note that a MTET can be treated as a parallel task without redundancy. Since we have discussed the FFCT of a parallel task in section 3.1, the FFCT of the zth MTET ($z = 1, 2, \ldots, Z$), denoted as x_z, can be derived from (4). Let X_z represent the random completion time of the zth MTET. From (12), the probability mass function (*pmf*) of X_z can be obtained as

$$\begin{cases}\Pr(X_z = x_z) = \Pr(F_z) \\ \Pr(X_z = \infty) = 1 - \Pr(F_z)\end{cases} \tag{13}$$

After finding all MTETs, the pmf of random completion time of the entire task (*i.e.*, T) can be derived by:

1) Sort Z MTETs in an increasing order of their FFCTs (*i.e.*, x_z), and make MTETs that have an identical FFCT constitute a group. Without loss generality, supposer there are C groups in total ($1 \leq C \leq Z$). Let x_c represent the FFCT of group c. After grouping the MTETs, the FFCTs of the groups are strictly increasing in c, *i.e.*, $0 < x_1 < \ldots < x_c < \ldots < x_C$. Denote E_c as the event that at least one of the MTETs in group c is available.
2) Let $p(x_c) = \Pr(T = x_c)$ represent the probability that random task completion time equals to the FFCT of group c. Since the task completion time T is the minimal completion time among the groups, we can first derive that
$$p(x_1) = \Pr(T = x_1) = \Pr(E_1) \tag{14}$$
3) The other probability for $T = x_c$ ($c = 2, 3, \ldots, C$) can be calculated by
$$p(x_c) = \Pr(T = x_c) = \Pr(E_c\overline{E}_{c-1}\overline{E}_{c-2}\ldots\overline{E}_1) \tag{15}$$
4) Finally, the probability that the task cannot be complete can be calculated by
$$p(\infty) = \Pr(T = \infty) = 1 - \sum_{c=1}^{C} p(x_c) \tag{16}$$

For calculating (14) and (15), we need further analyze the MTETs in the groups. Suppose there are Z_c MTETs in group c ($1 \leq Z_c \leq Z$, $\sum_{c=1}^{C} Z_c = Z$), and those MTETs are numbered in an arbitrary order $1, \ldots, i, \ldots, Z_c$. Now, event E_c can be expressed as

$$E_c = \bigcup_{i=1}^{Z_c} F_i, \quad c = 1, 2, \ldots, C \tag{17}$$

Submit (17) into (14) and (15), we can obtain

$$\begin{aligned} p(x_1) &= \Pr(\bigcup_{i=1}^{Z_1} F_i) \\ p(x_c) &= \Pr(\bigcup_{i=1}^{Z_c} F_i\overline{E}_{c-1}\overline{E}_{c-2}\cdots\overline{E}_1), \quad c = 2, \ldots, C \end{aligned} \tag{18}$$

Using the Bayesian theorem on conditional probability, (18) can be written as

$$\begin{aligned} p(x_1) &= \Pr(F_1) + \sum_{i=2}^{Z_1}\Pr(F_i)\cdot\Pr(\overline{F}_{i-1}\cdots\overline{F}_1|F_i) \\ p(x_c) &= \sum_{i=1}^{Z_c}\Pr(F_i) \\ &\quad \cdot\Pr(\overline{F}_{i-1}\cdots\overline{F}_1\overline{E}_{c-1}\cdots\overline{E}_1|F_i), \quad c = 2, 3, \ldots, C \end{aligned} \tag{19}$$

Note that $\overline{E}_c$ is that all MTETs in group c are failed simultaneously. Therefore, event $\overline{F}_{i-1}\cdots\overline{F}_1\overline{E}_{c-1}\cdots\overline{E}_1$ in (19) can be finally translated to the event that multiple MTETs are failed simultaneously. For simplification, let $s = (i-1) + Z_{c-1} + \cdots + Z_1$. Then, $\Pr(\overline{F}_{i-1}\cdots\overline{F}_1\overline{E}_{c-1}\cdots\overline{E}_1|F_i)$ can be written as $\Pr(\overline{F}_s\cdots\overline{F}_1|F_i)$, which has a same form with $\Pr(\overline{F}_{i-1}\cdots\overline{F}_1|F_i)$ essentially.

Now, two critical forms of probability in (19) need to be calculated, *i.e.*, $\Pr(F_i)$ and $\Pr(\overline{F}_s \cdots \overline{F}_1|F_i)$. $\Pr(F_i)$ can be derived from (12). However, another important probability, *i.e.*, $\Pr(\overline{F}_s \cdots \overline{F}_1|F_i)$ is hard to calculated directly, as some elements (including subtasks, servers and links) may belong to multiple MTETs. For calculating $\Pr(\overline{F}_s \cdots \overline{F}_1|F_i)$, we can use

$$\begin{aligned}
&\Pr(\overline{F}_s \cdots \overline{F}_1|F_i) \\
&\quad = 1 - \Pr(F_s \bigcup F_{s-1} \bigcup \cdots \bigcup F_1|F_i) \\
&\quad = 1 - \sum_{j=1}^{s} \Pr(F_j|F_i) + \sum_{j,k}^{j\neq k} \Pr(F_jF_k|F_i) \\
&\qquad - \sum_{j,k,l}^{j\neq k\neq l} \Pr(F_jF_kF_l|F_i) + \cdots + (-1)^s \Pr(F_s \cdots F_1|F_i)
\end{aligned} \tag{20}$$

As seen in (20), the calculation of probabilities $\Pr(F_j|F_i)$, $\Pr(F_jF_k|F_i)$, $\Pr(F_jF_kF_l|F_i)$, ..., $\Pr(F_s \cdots F_1|F_i)$ is more easily than the direct calculation of $\Pr(\overline{F}_s \cdots \overline{F}_1|F_i)$. If we can find the critical element sets resulting in events $F_j|F_i$, $F_jF_k|F_i$, ..., $F_s \cdots F_1|F_i$, those probabilities can be easily derived from (12).

Suppose MTET *i* and *j* have element sets ψ_i and ψ_j, respectively. Let $\psi_{j|i}$ represent the critical element set that does not affect the run of MTET *i*, but decides if MTET *j* is complete. The pseudocode for deriving $\psi_{j|i}$ from ψ_j and ψ_i is shown in Algorithm 2. As seen in line 6-11 of algorithm 2, when element e_d belongs to ψ_j and ψ_i, it cannot be failed during the first time interval t_i for guaranteeing the completion of MTET *i*, but it may be failed during the following time interval $t_j - t_i$. Therefore, $(e_d, t_j - t_i)$ is a critical element of $\psi_{j|i}$.

Suppose $\psi_{jk|i}$ is the set of the critical elements that do not affect the run of MTET *i*, but decide if event $F_jF_k|F_i$ happens. Let '$\vee$' represent the operation that derive $\psi_{jk|i}$ from $\psi_{j|i}$ and $\psi_{k|i}$ (*i.e.*, $\psi_{jk|i} = \psi_{j|i} \vee \psi_{k|i}$). The pseudocode for deriving $\psi_{jk|i}$ is shown in Algorithm 3. Now, the critical element set $\psi_{j\cdots k|i}$ resulting in event $F_j \cdots F_k|F_i$ can be obtained as $\psi_{j\cdots k|i} = \psi_{j|i} \vee \cdots \vee \psi_{k|i}$ by using algorithm 3 iteratively.

Note that $\psi_{j\cdots k|i}$ is an element set essentially. Therefore, probability $\Pr(F_j \cdots k|i)$ can be obtained as all elements in $\psi_{j\cdots k|i}$ do not failed during their FFCTs, *i.e.*,

$$\begin{aligned}
\Pr(F_{j\cdots k|i}) = \Pr(\psi_{j\cdots k|i}) &= \prod_{d=1}^{|\psi_{j\cdots k|i}|} \Pr(e_d) \\
&= \prod_{d=1}^{|\psi_{j\cdots k|i}|} \exp\big(-\zeta_d \cdot t(e_d)\big)
\end{aligned} \tag{21}$$

Finally, from (19),(20),(21) and (16), the pmf of random task completion time (*i.e.*, $p(x_c) = \Pr(T = x_c)$) can be derived.

Algorithm 2 Derive the Critical Elements Belong to $\psi_{j|i}$

```
Require: $\psi_j, \psi_i$
Ensure: $\psi_{j|i}$
 1: $\psi_{j|i} \leftarrow \emptyset$
 2: for $d = 1$ to $|\psi_j|$ do
 3:     get element $(e_d, t(e_d))$ from $\psi_j$, $t_j \leftarrow t(e_d)$
 4:     if $e_d$ cannot be found in $\psi_i$ then
 5:         let $(e_d, t_j) \in \psi_{j|i}$
 6:     else
 7:         get element $(e_d, t(e_d))$ from $\psi_i$, $t_i \leftarrow t(e_d)$
 8:         if $t_j > t_i$ then
 9:             let $(e_d, t_j - t_i) \in \psi_{j|i}$
10:         end if
11:     end if
12: end for
13: return $\psi_{j|i}$
```

Algorithm 3 Derive the Critical Elements Belong to $\psi_{jk|i}$

```
Require: $\psi_{j|i}, \psi_{k|i}$
Ensure: $\psi_{jk|i}$
 1: $\psi_{jk|i} \leftarrow \emptyset$
 2: for $d = 1$ to $|\psi_{j|i}|$ do
 3:     get element $(e_d, t(e_d))$ from $\psi_{j|i}$, $t_{j|i} \leftarrow t(e_d)$
 4:     if $e_d$ can be found in $\psi_{k|i}$ then
 5:         get element $(e_d, t(e_d))$ from $\psi_{k|i}$, $t_{k|i} \leftarrow t(e_d)$
 6:         let $(e_d, \max(t_{j|i}, t_{k|i})) \in \psi_{jk|i}$
 7:         delete $(e_d, t_{k|i})$ from $\psi_{k|i}$
 8:     else
 9:         let $(e_d, t_{j|i}) \in \psi_{jk|i}$
10:     end if
11: end for
12: for $d = 1$ to $|\psi_{k|i}|$ do
13:     let $(e_d, t(e_d)) \in \psi_{jk|i}$
14: end for
15: return $\psi_{jk|i}$
```

4) PERFORMABILITY METRIC

Having the pmf of T, we can further evaluate the performability metric for executing the entire task. The performability metric can be treated as a precise evaluation of random performance with considering the important reliability-performance correlation [5]. The reliability of the entire task can be defined as the probability that it is complete, that is,

$$R = 1 - \Pr(T = \infty) \tag{22}$$

Apparently, the completion time is of critical important to a parallel task. However, since there exists the special case that the task is failed (*i.e.*, $T = \infty$), we cannot directly derive the expected completion time as the performability metric. In this paper, we use the inverse of the task completion time to quantify random performance (denoted by V). That is, $V = 1/T$, and $V = 0$ if $T = \infty$. This is rational since there no performance can be gained if the task cannot be complete.

Now, the performability metric, denoted by I, can be defined as the expected value of random variable V, which is written as

$$I = \mathrm{E}(V) = \sum_{x \neq \infty} \frac{1}{x_c} \cdot p(x_c) \tag{23}$$

IV. OPTIMIZATION TECHNIQUE

A. OPTIMIZATION MODEL

Definitely, adding a copy of any subtask has a positive effect on improving the reliability of completing the task, which also implies that it can improve the performability of the task. However, an excessive number of redundant copies may not be rational. This is because that, with the increase of the number of redundant copies, the increment in the performability metric caused by adding a copy becomes increasingly slight. Those excessive copies occupy a large amount of resources, which results in an inefficient resource utilization manner, especially for the network system of which resources need to be shared by a large amount of tasks. Let M_r represent the number of redundant copies created for executing the task. It can be obtained as

$$M_r = \sum_{m=1}^{M} |\phi_m| - M \tag{24}$$

where ϕ_m $(m = 1, 2, \ldots, M)$ is assignment strategy of subtask m. For example, in Fig. 3, $\phi_4 = \{1, 5\}$ means subtask 4 is assigned to server 1 and 5.

For avoiding the situation that a task occupies excessive resources, the system can give a threshold value of M_r, which represents the maximal number of redundant copies that can be created for executing the task. Denote the threshold as $\overline{M}_r$. Inequality $M_r \leq \overline{M}_r$ should be held when the network system executes the parallel task. For finding a task execution strategy that maximizes the performability metric, we also need to make decision on how to distribute the redundant copies to the subtasks and how to assign those subtasks and copies to servers, which can be described by the following optimization model:

$$\begin{aligned} &\text{Decision Variable}: \phi_1, \phi_2, \ldots, \phi_M \\ &\text{Objective}: \max I = \sum_{x \neq \infty} \frac{1}{x_c} \cdot p(x_c) \\ &\qquad \text{s.t. } \phi_m \neq \emptyset, \phi_m \subseteq \Theta \quad (m = 1, 2, \ldots, M) \\ &\qquad M_r = \sum_{m=1}^{M} |\phi_m| - M \leq \overline{M}_r \\ &\qquad u_n \leq 1 - \overline{u}_n \quad (n = 1, 2, \ldots, N) \end{aligned} \tag{25}$$

where $\Theta = \{1, 2, \ldots, N\}$ is the set of all available servers in the network system. $\overline{u}_n$ and u_n are the resource utilization of nth server before and after hosting the subtasks, respectively.

5) *Crossover operator*: Since a chromosome consists of M binary substrings, a uniform crossover operator [20] is used to make each substring have a chance to be covered by the crossover operator. As shown in Fig. 5, a binary

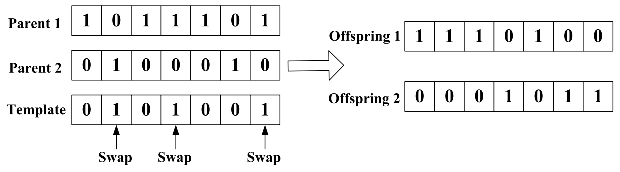

FIGURE 5. Uniform crossover operator.

template having the same length with the chromosome is first randomly generated. Then, materials of parents at same bit positions are swapped according to information of template at the corresponding positions. For example, in Fig. 5, '1' means swap, and materials in parents remain unchange otherwise.

6) *Mutation operator*: It randomly selects M bits to change from '1(0)' to '0(1)', which has a positive effect on keeping population diversity. 7) *Elitism strategy*: The best chromosome of the population may fail to produce offspring in the next generation. The elitism strategy that copies the best chromosome into the succeeding generation can effectively solve this problem.

The GA randomly generates K chromosomes forming the initial population. The crossover rate and mutation rate can be set within the ranges of 0.85-0.95 and 0.01-0.05. respectively. The GA is terminated when the best chromosome remains unchanged for a certain number of generations, or when the number of generations reaches the maximum.

V. EXAMPLES

To begin with experiments, the parameters need to be estimated first. The processing speed of a server can be measured by mapping the CPU frequency onto the million instructions per second (MIPS) rating. The reliability parameters (*i.e.*, the failure rates of subtasks, servers and links) can be obtained by letting the CN accumulates two indices: the total running time (τ) and the number of failures (σ) of individual subtasks, servers and links. Then, according to the maximum likelihood estimation (MLE), the CN can estimate the individuals' failure rates by calculating σ/τ [4]. The network structure of our experimental scenario is shown in Fig. 3, which also gives the work requirements of the subtasks (w_n), the computational abilities of the servers (f_n), and the failure rates of the subtasks, servers and links ($\lambda_m, \theta_n, \mu_k$). The precedence constraints of the subtasks are shown in Fig. 2.

A. EVALUATION OF THE PERFORMABILITY METRIC

From (1), we can first derive the FFET of each subtask. For example, the FFET of subtask 3 hosted on server 1 is $u_{31} = 16$ s. The FFETs of all subtasks are shown in Fig. 3. But note that, if a subtask has predecessor subtasks, its FFCT is the sum of the FFET and FFWT, *i.e.*, $t_m = u_m + v_m$.

According to the task execution strategy shown in Fig. 3 (*i.e.*, $\phi_1 = \{2, 3\}, \phi_2 = \{3\}, \phi_3 = \{1, 4\}, \phi_4 = \{1, 5\}, \phi_5 = \{3\}$), we can found that there are eight MTETs in total ($Z = 8$). The information of all possible MTETs are listed in TABLE 1. As for any one MTET, it must and only have subtask 1, 2, 3, 4 and 5. Therefore, the FFCTs of all subtasks in the MTET can be derived by using algorithm 1, which

TABLE 1. **Information of all possible MTETs.**

No.	FFCT	Element set of MTET expressed in the two-field record form (ψ_z)
1	77 s	$(A_{12},61)(A_{23},15)(A_{31},31)(A_{41},57)(A_{53},77)\vert(S_2,61)(S_3,77)(S_1,57)\vert(L_1,77)(L_4,61)(L_7,61)(L_8,77)(L_3,57)$
2	77 s	$(A_{12},61)(A_{23},15)(A_{31},31)(A_{45},51)(A_{53},77)\vert(S_2,61)(S_3,77)(S_1,31)(S_5,77)\vert(L_1,77)(L_4,61)(L_7,61)(L_8,77)(L_3,31)(L_2,51)(L_6,51)$
3	71 s	$(A_{12},55)(A_{23},15)(A_{34},25)(A_{41},51)(A_{53},71)\vert(S_2,55)(S_3,71)(S_4,25)(S_1,51)\vert(L_1,71)(L_4,55)(L_7,55)(L_8,71)(L_2,25)(L_5,25)(L_3,51)$
4	71 s	$(A_{12},55)(A_{23},15)(A_{34},25)(A_{45},45)(A_{53},71)\vert(S_2,55)(S_3,71)(S_4,25)(S_5,45)\vert(L_1,71)(L_4,55)(L_7,55)(L_8,71)(L_2,45)(L_5,25)(L_3,45)$
5	61 s	$(A_{13},45)(A_{23},15)(A_{31},31)(A_{41},57)(A_{53},61)\vert(S_3,61)(S_1,57)\vert(L_1,61)(L_4,61)(L_8,61)(L_3,57)$
6	61 s	$(A_{13},45)(A_{23},15)(A_{31},31)(A_{45},51)(A_{53},61)\vert(S_3,61)(S_1,31)(S_5,51)\vert(L_1,61)(L_4,61)(L_8,61)(L_3,31)(L_2,51)(L_6,51)$
7	55 s	$(A_{13},39)(A_{23},15)(A_{34},25)(A_{41},51)(A_{53},55)\vert(S_3,55)(S_4,25)(S_1,51)\vert(L_1,55)(L_4,55)(L_8,55)(L_2,25)(L_5,25)(L_3,51)$
8	55 s	$(A_{13},39)(A_{23},15)(A_{34},25)(A_{45},45)(A_{53},55)\vert(S_3,55)(S_4,25)(S_5,45)\vert(L_1,55)(L_4,55)(L_8,55)(L_2,45)(L_5,25)(L_6,45)$

also takes the precedence constraints into account. Then we can get the element set of the MTET of which elements are extended to the two-field record form, *i.e.*, the third column in TABLE 1. The FFCTs of all the MTETs can be derived from (4), which are listed in the second column in TABLE 1. Meanwhile, the completion probability of the MTET (*i.e.*,$\Pr(F_z)$) can be calculated from (12). For instance, for the METE 5 shown in TABLE 1, its completion probability is

$$\Pr(F_5) = \Pr(\psi_5) = e^{-\lambda_1 \cdot 45} e^{-\lambda_2 \cdot 15} e^{-\lambda_3 \cdot 31} e^{-\lambda_4 \cdot 57} e^{-\lambda_5 \cdot 61} \\ e^{-\theta_3 \cdot 61} e^{-\theta_1 \cdot 57} e^{-\eta_1 \cdot 61} e^{-\eta_4 \cdot 61} e^{-\theta_8 \cdot 61} e^{-\eta_3 \cdot 57} = 0.6036 \tag{26}$$

According to the FFCTs of eight MTETs, we can divide the MTETs into four groups with the increasing order of identical FFCTs, that is,

$$\begin{cases} G_1 = \{\psi_7, \psi_8\} \ \text{with } x_1 = 55 \\ G_2 = \{\psi_5, \psi_6\} \ \text{with } x_2 = 61 \\ G_3 = \{\psi_3, \psi_4\} \ \text{with } x_3 = 71 \\ G_4 = \{\psi_1, \psi_2\} \ \text{with } x_4 = 77 \end{cases} \tag{27}$$

For deriving the pmf of random task completion time T, we first calculate $p(x_1) = \Pr(T = x_1) = \Pr(E_1)$. Since $E_1 = F_7 + F_8$, from (19), $\Pr(E_1)$ can be written as

$$\begin{aligned} \Pr(E_1) &= \Pr(F_7) + \Pr(\overline{F}_7 F_8) \\ &= \Pr(F_7) + \Pr(F_8)\Pr(\overline{F}_7|F_8) \end{aligned} \tag{28}$$

where $\Pr(F_7)$ and $\Pr(F_8)$ can be obtained from (12). To calculate $\Pr(\overline{F}_7|F_8) = 1 - \Pr(F_7|F_8) = 1 - \Pr(\psi_{7|8})$, we need identify the critical elements that belong to $\psi_{7|8}$. According to algorithm 2, the critical elements in $\psi_{7|8}$ are $(A_{41}, 51)$, $(S_1, 51)$ and $(L_3, 51)$. From (21), probability $\Pr(\psi_{7|8})$ can be obtained as

$$\Pr(\psi_{7|8}) = e^{-\lambda_4 \times 51} \cdot e^{-\theta_1 \times 51} \cdot e^{-\mu_3 \times 51} = 0.8494 \tag{29}$$

Substituting (29) into (28), $\Pr(E_1)$ is calculated as 0.7070. Then, from (17), $p(x_2) = \Pr(T = x_2) = \Pr(E_2\overline{E}_1)$ can be further calculated by

$$\begin{aligned} \Pr(E_2\overline{E}_1) &= \Pr(F_5\overline{E}_1) + \Pr(F_6\overline{E}_1\overline{F}_5) \\ &= \Pr(F_5)\Pr(\overline{E}_1|F_5) + \Pr(F_6)\Pr(\overline{E}_1\overline{F}_5|F_6) \\ &= \Pr(F_5)\Pr(\overline{F}_8\overline{F}_7|F_5) + \Pr(F_6)\Pr(\overline{F}_8\overline{F}_7\overline{F}_5|F_6) \end{aligned} \tag{30}$$

In (30), $\Pr(\overline{F}_8\overline{F}_7|F_5)$ and $\Pr(\overline{F}_8\overline{F}_7\overline{F}_5|F_6)$ are hard to be calculated directly. Therefore, we can apply the method given by (20). For example, $\Pr(\overline{F}_8\overline{F}_7\overline{F}_5|F_6)$ can be derived by

$$\begin{aligned} &\Pr(\overline{F}_8\overline{F}_7\overline{F}_5|F_6) \\ &\quad = 1 - \Pr(F_8 \bigcup F_7 \bigcup F_5|F_6) \\ &\quad = 1 - \Pr(F_8|F_6) - \Pr(F_7|F_6) - \Pr(F_5|F_6) \\ &\qquad + \Pr(F_8F_7|F_6) + \Pr(F_8F_5|F_6) + \Pr(F_7F_5|F_6) \\ &\qquad - \Pr(F_8F_7F_5|F_6) \end{aligned} \tag{31}$$

For calculating the conditional probabilities in (30), the corresponding critical element sets need to be found first. Take the calculation of $\Pr(F_8F_7F_5|F_6)$ as an example. The critical element set that results in event $F_8F_7F_5|F_6$ is $\psi_{875|6}$. Using algorithm 2, we can first obtain $\psi_{8|6}$, $\psi_{7|6}$ and $\psi_{5|6}$ as

$$\begin{aligned} \psi_{8|6} &= \{(A_{34}, 25), (S_4, 25), (L_5, 25)\} \\ \psi_{7|6} &= \{(A_{34}, 25), (A_{41}, 51), (S_4, 25), \\ &\qquad (S_1, 20), (L_5, 25), (L3, 20)\} \\ \psi_{5|6} &= \{(A_{41}, 57), (S_1, 26), (L_3, 26)\} \end{aligned} \tag{32}$$

Then, $\psi_{875|6}$ can be further derived by using algorithm 3 iteratively, that is,

$$\begin{aligned} &\psi_{875|6} \\ &= \psi_{8|6} \vee \psi_{7|6} \vee \psi_{5|6} \\ &= \{(A_{34}, 25), (A_{41}, 57), (S_4, 25), (S_1, 26), (L_5, 25), (L_3, 26)\} \end{aligned} \tag{33}$$

From (21), $\Pr(F_8F_7F_5|F_6) = \Pr(\psi_{875|6}) = 0.8237$. After obtaining $\Pr(F_8F_7|F_6)$, $\Pr(F_8F_5|F_6)$ and $\Pr(F_7F_5|F_6)$ in a similar way, those values can be substituted into (31), and $\Pr(\overline{F}_8\overline{F}_7\overline{F}_5|F_6)$ is calculated as 0.0069.

Adopting the method given by (31), (32) and (33), all conditional probabilities in (30) can be obtained, and probability $\Pr(T = 61) = \Pr(E_2\overline{E}_1)$ is evaluated as 0.0420. Similarly, $\Pr(T = 71)$ and $\Pr(T = 77)$ are derived as 0.0299 and 0.0020, respectively. Finally, from (16), $\Pr(T = \infty)$ is calculated as 0.2191. The pmf of random task completion time T is illustrated in TABEL 2.

For verifying the correction of the proposed theoretical model, we also design a simulation experiment based on the *Monte Carlo* method, which simulates the entire execution process of the task. Fig. 6 shows that the comparison between the theoretical results and statistical results

TABLE 2. The PMF of random task completion time.

x_c	55	61	71	77	∞
$p(x_c)$	0.7070	0.0420	0.0299	0.0020	0.2191

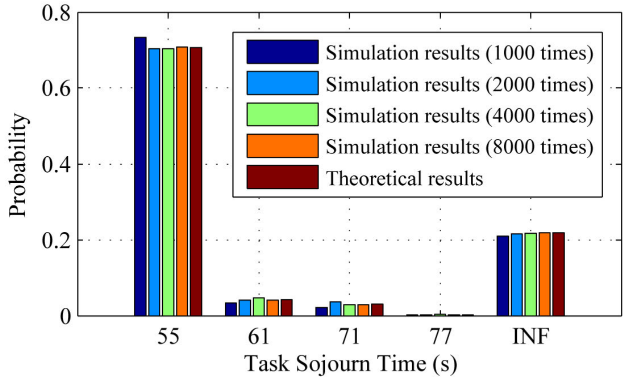

FIGURE 6. Statistical results vs. theoretical results at different times of simulation.

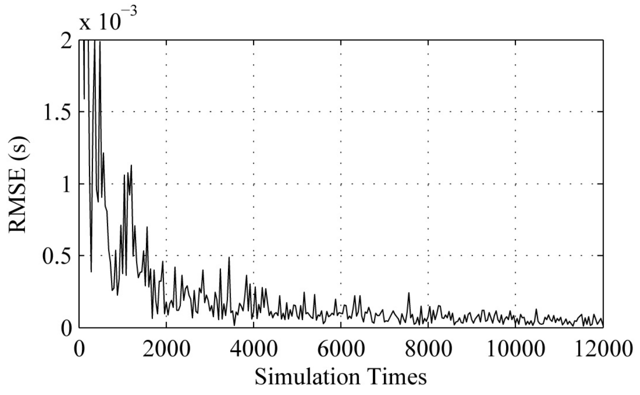

FIGURE 7. RMSEs of simulation results at different simulation times.

derived by running the simulation program 1000, 2000, 4000 and 8000 times. Fig. 7 illustrates root-mean-square errors (RMSE) of simulation results derived by running the simulation program at different times. As shown in the figures, the statistical results of the pmf of random task completion time are very close to the theoretical results evaluated by the proposed model, which witness that our theoretical model is justified.

Now, from (23), the performability metric of the entire task executed by the given resource assignment strategy can be obtained as

$$I = \frac{1}{55} \times 0.7070 + \frac{1}{61} \times 0.0420 + \frac{1}{71} \times 0.0299 + \frac{1}{77} \times 0.0020 = 13.9902 \times 10^{-3}\ (\mathrm{s}^{-1}) \quad (34)$$

The statistical results of the performability metric gained by 300 experiments are shown in Fig. 8. Each experiment runs the simulation program 1000 times and then estimates the performance metric. It can be observed that the statistical results only fluctuate around the theoretical result

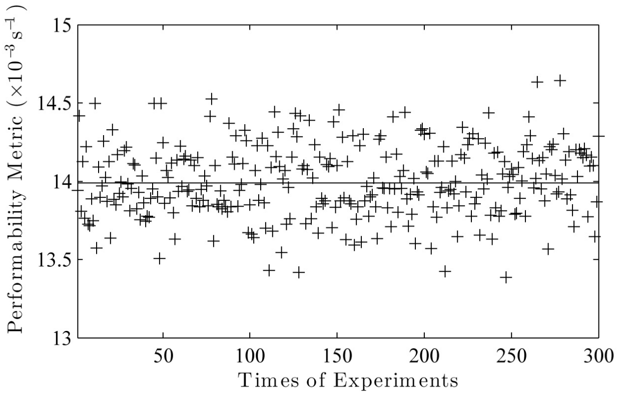

FIGURE 8. Statistical results of the performability metric derived by 300 experiments.

(*i.e.*, the line shown in Fig. 8), which also verifies the proposed correlation model.

B. OPTIMIZATION OF THE PERFORMABILITY METRIC

TABLE 3 lists some representative task execution strategies to demonstrate how the performability metric is affected by various factors. Those factors mainly include:

1) *Distribution of redundant copies to subtasks*

Strategies 1, 2, 3 and 4 listed in TABLE 3 assign five redundant copies in total for executing the task (*i.e.*,$\overline{M}_r = 5$), but make those copies distribute to different subtasks. There are two, three, four and five subtasks have redundant copies in strategies 1, 2, 3 and 4, respectively. It can be found that even though those strategies assign the identical number of redundant copies to the subtasks in total, but their performability metrics are totally different, *i.e.*, $I_4 > I_3 > I_2 > I_1$. In general, if the distribution of redundant copies is more 'balance' (*i.e.*, make all subtasks have redundant copies as far as possible), the performability may be better. This is because that failures of each subtask inevitably results in that the entire task cannot be complete. Thus, enhancing the reliability of all subtasks is superior to emphasizing the reliability of partial subtasks.

2) *Precedence Constraint and Computational Speed*

We also list the strategy described in Section 5.1 in TABLE 3 (*i.e.*, strategy 5). It can be found that, although there just are three subtasks have redundant copies in strategy 5, its performability metric are superior to strategy 3 in which four subtasks have redundant copies. This is mainly because that important precedent subtasks including subtask 1, 2 and 3 are assigned to server 3 and 4 of which processors have relatively high core frequencies. Compare to the other servers, server 3 and 4 have higher computational speeds. Therefore, if the network system assigns precedent subtasks to faster server, those subtasks can have shorter FFETs, which also implies that they can have lower probabilities to be failed. Since the execution of precedent subtasks directly affect the execution of successor subtasks, assigning precedent subtasks to faster and more reliable servers may have more significant effect on improving the performability metric.

TABLE 3. Comparison of the performability metric of different task execution strategies.

No.	Task execution strategy	M_r	$I\ (\times 10^{-3}\ \text{s}^{-1})$	Description of task execution strategy
1	$\phi_1 = \{1\}, \phi_2 = \{2\}, \phi_3 = \{3\}, \phi_4 = \{3,4\}, \phi_5 = \{1,2,3,4,5\}$	5	8.1465	Two subtasks have redundant copies
2	$\phi_1 = \{1\}, \phi_2 = \{2\}, \phi_3 = \{3,5\}, \phi_4 = \{1,4\}, \phi_5 = \{2,3,4,5\}$	5	8.9299	Three subtasks have redundant copies
3	$\phi_1 = \{1\}, \phi_2 = \{2,5\}, \phi_3 = \{3,5\}, \phi_4 = \{1,5\}, \phi_5 = \{2,3,4\}$	5	12.7353	Four subtasks have redundant copies
4	$\phi_1 = \{1,5\}, \phi_2 = \{2,3\}, \phi_3 = \{3,4\}, \phi_4 = \{4,5\}, \phi_5 = \{2,3\}$	5	14.8568	Five subtasks have redundant copies
5	$\phi_1 = \{2,3\}, \phi_2 = \{3\}, \phi_3 = \{1,4\}, \phi_4 = \{1,5\}, \phi_5 = \{3\}$	3	13.9902	The example described in Section 5.1
6	$\phi_1 = \{4\}, \phi_2 = \{3,5\}, \phi_3 = \{1,4\}, \phi_4 = \{3,5\}, \phi_5 = \{3,5\}$	4	15.7041	The optimal strategy given threshold $\overline{M}_r = 4$

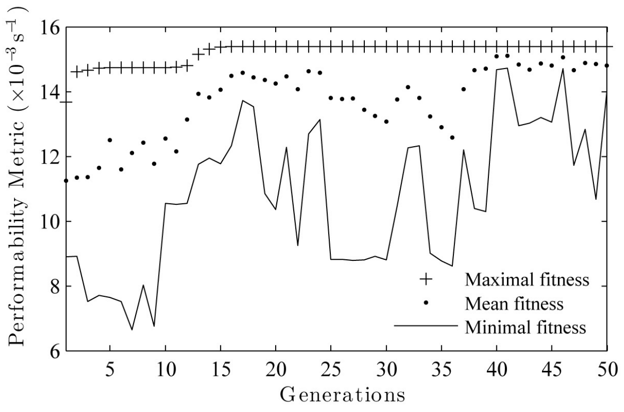

FIGURE 9. Runs of the genetic algorithm.

Due to those complicated factors, the design of a rational task execution strategy is usually a NP-hard problem. The optimization technique presented in Section 4 can be adopted to find an optimal task execution strategy.

Suppose the maximal number of allowed redundant copies is $\overline{M}_r$. According to our experiment situation, the parameters of the genetic algorithm described in Section 4.2 is set as: population size 20, generation 50, crossover rate 0.95, and mutation rate 0.05. Fig. 9 shows the run of the genetic algorithm. The maximal fitness, mean fitness, minimal fitness of each generation are shown in the graph. The minimal fitness shows that the genetic algorithm can keep the diversity of the population. Given threshold $\overline{M}_r = 4$, the best performability metric is obtained as $15.7041 \times 10^{-3}\ \text{s}^{-1}$, and the corresponding task execution strategy is $\phi_1 = \{4\}$, $\phi_2 = \{3,5\}$, $\phi_3 = \{1,4\}$, $\phi_4 = \{3,5\}$ and $\phi_5 = \{3,5\}$ (*i.e.*, strategy 6 shown in TABLE 3). It can be found that server 2 does not host any subtasks in the optimal task execution strategy. This is mainly because that it has the worst computational speed among five servers. Meanwhile, the distribution of redundant copies to the subtasks are balance and the precedent subtasks (*i.e.*, subtask 1, 2 and 3) have been assigned to servers that have high core frequencies (*i.e.*, server 3 and 4). This proves our discussion mentioned in Section 5.2.

VI. LITERATURE REVIEW AND DISCUSSION

In this paper, we first proposed a theoretical model for evaluating the performability metric of a redundant parallel task in the network. Theoretical modeling is always a critical research field that can further contribute to designing a rational resource assignment strategy. There are many studies focused on building theoretical models for analyzing reliability or performance of the network system. For example, Jung *et al.* applied a performance model emphasizing network delay for a parallelized task [21]. Ke *et al.* also proposed a performance model considering network topology, network traffic, and data size for a parallelized task processed by the MapReduce mechanism [22]. Mo *et al.* proposed a reliability model for analyzing dependent propagated failures existed in the network system [23]. Bahaga and Madisetti presented a reliability model capturing numerous failures of a Hadoop system with a large number of machines [24]. However, those models only focus on a single metric and do not consider the complicated reliability-performance (R-P) correlation.

Considerable recent studies have gradually concerned on joint modeling for correlating performance and reliability. For example, Tudoran *et al.* investigated the R-P correlation of a network system for achieving fast response of data transmission and alleviating adverse effects caused by low reliability and reusability [25]. Ghosh *et al.* quantified the R-P correlation as performability metrics, including expected response time, expected execution time, and expected delay time, for a cloud service [7]. Kwiatkowska *et al.* proved that probabilistic models, such as continuous-time Markov chains and Markov reward models, can be used to study the R-P correlation of different stochastic systems [26]. Machida *et al.* adopted a Markov regenerative process to analysis the performability of a distributed RAID storage system [27]. Those researches propose a flexible approach that integrates interacting reliability and performance sub-models to reduce the complexity of correlation modeling. In general, such a method first needs to extract some interaction factors not only decided by reliability but also affecting performance as critical conditional variables to build reliability and performance sub-models, such as our prior researches [9], [28]. However, this method is not suitable for a redundant parallel task in the network system since it is hard to exactly find interaction factors due to the complicated network structure [29]. One of our recent work presented a fine-grained R-P correlation model that quantifies random change of performance caused by hardware/software failures for a single task, but it did not consider parallel and redundant computing pattern [30]. Another our recent work has also investigated resource optimization for a big data task with considering the R-P correlation [18], but it does not consider important failure correlations existed among multiple types of failures. In this work, we present a new R-P correlation model for a redundant parallel task in the network system, which systemically analyzes not only task failures, server failures, and link failures but also complicated failure correlations.

Another important research filed related to this paper is resource optimization. Many optimization techniques such as online optimization techniques [31] and forecast techniques [10] are usually applied for a dynamic application environment, but not for a redundant parallel task in the network system. As for the situation that finding an optimal resource optimization strategy from a specific optimization model, using heuristics algorithms to search optimal solutions is an efficient approach. Our prior work presented a novel resource scheduling mechanism based on the bionic nervous system to solve a R-P correlation optimization model for maximizing the net profit of a cloud system [32], which is designed from the system perspective but not the task perspective. Therefore, it cannot be directly applied for the redundant parallel task. There are many researches focus on improving performance of tasks, such as reducing average task completion time [33], minimizing service delay [34]. Meanwhile, considerable researches have adopted various heuristic algorithms (such as the ant colony algorithm [35], the immune algorithm [36], the genetic algorithm [37], and the particle swarm algorithm [38]) to search optimal redundancy strategies to maximize reliability. Those researches prove that the heuristic algorithm is a feasible and effective approach to solve the complicated optimization problem about redundancy assignment and allocation. However, those existed optimal techniques does not address the important R-P correlation. Therefore, they cannot meet reliability and performance requirements simultaneously. The optimization techniques described in this paper are on the proposed R-P correlation model, and can effectively solve the issue.

VII. CONCLUSION

Over the last few years, the network system has been widely deployed in multiple application fields, such as scientific computing, large-scale data analysis, and distributed service. To efficiently and successfully execute a redundant parallel task in the network system, performance and reliability should be considered simultaneously.

The work presented in this paper is original in that it systemically studies the complicated R-P correlation for a redundant parallel task in the network system. We present a pertinence modeling approach to build the R-P correlation model for the network system. Different from some existed models, our model captures representative features of the network system, typically, multiple types of failures including task failures, server failures, link failures and complicated failure correlations. Taking those comprehensive reliability factors into account can indeed improve the fidelity of the model, but it also increases the complexity of calculating the performability metric. Therefore, we also propose an algorithm encompassing the extended tree topology, minimal task execution trees, and the Bayesian approach to evaluate the performability metric. Finally, we also study the corresponding optimization techniques for maximizing the performability metric. A genetic algorithm is implemented to obtain an optimal strategy that maximizes the expected performance of a task executed in the parallel and redundant manner.

The numerical examples in this work illustrate that the process of deriving the performability metric from the proposed R-P correlation model. The examples also show that there are many important factors seriously affect the performability metric, such as precedence constraints existed in subtasks, distribution of redundant subtasks, and computational speeds of host servers. However, our optimization technique can effectively find an optimal resource assignment strategy maximizing the performability metric.

Due to various application environments, the application demands the network system may become more complicated. For example, not only reliability and performance, but also energy consumption may be considered simultaneously. Extending the proposed model to more complicated R-P-E (reliability-performance-energy) correlation model and developing new optimization techniques to balance the reliability-energy and performance-energy tradeoffs will be studied in our future work.

REFERENCES

[1] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, "Cloud migration: A case study of migrating an enterprise IT system to IaaS," in *Proc. IEEE 3rd Int. Conf. Cloud Comput.*, Jul. 2010, pp. 450–457.

[2] Y. Jiang, "A survey of task allocation and load balancing in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 585–599, Feb. 2016.

[3] R. Entezari-Maleki, K. S. Trivedi, and A. Movaghar, "Performability evaluation of grid environments using stochastic reward nets," *IEEE Trans. Depend. Sec. Comput.*, vol. 12, no. 2, pp. 204–216, Mar. 2015.

[4] M. Xie, Y. Dai and K. L. Poh, *Computing Systems Reliability: Models and Anaysis*. New York, NY, USA: Kluwer, 2004.

[5] J. F. Meyer, "On evaluating the performability of degradable computing systems," *IEEE Trans. Comput.*, vols. C–29, no. 8, pp. 720–731, Aug. 1980.

[6] K. R. Pattipati, Y. Li, and H. A. P. Blom, "A unified framework for the performability evaluation of fault-tolerant computer systems," *IEEE Trans. Comput.*, vol. 42, no. 3, pp. 312–326, Mar. 1993.

[7] R. Ghosh, K. S. Trivedi, V. K. Naik, and D. S. Kim, "End-to-end performability analysis for infrastructure-as-a-service cloud: An interacting stochastic models approach," in *Proc. IEEE 16th Pacific Rim Int. Symp. Dependable Comput.*, Dec. 2010, pp. 125–132.

[8] B. Yang, F. Tan, and Y.-S. Dai, "Performance evaluation of cloud service considering fault recovery," *J. Supercomput.*, vol. 65, no. 1, pp. 426–444, Jul. 2013.

[9] X. Qiu, Y. Dai, Y. Xiang, and L. Xing, "A hierarchical correlation model for evaluating reliability, performance, and power consumption of a cloud service," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 46, no. 3, pp. 401–412, Mar. 2016.

[10] J. Subirats and J. Guitart, "Assessing and forecasting energy efficiency on cloud computing platforms," *Future Gener. Comput. Syst.*, vol. 45, pp. 70–94, Apr. 2015.

[11] M. A. Rahman, M. H. Manshaei, E. Al-Shaer, and M. Shehab, "Secure and private data aggregation for energy consumption scheduling in smart grids," *IEEE Trans. Depend. Sec. Comput.*, vol. 14, no. 2, pp. 221–234, Mar./Apr. 2017.
[12] F. Ramezani, J. Lu, and F. K. Hussain, "Task-based system load balancing in cloud computing using particle swarm optimization," *Int. J. Parallel Program.*, vol. 42, no. 5, pp. 739–754, 2013.
[13] Y. Han, J. Chan, T. Alpcan, and C. Leckie, "Using virtual machine allocation policies to defend against co-resident attacks in cloud computing," *IEEE Trans. Depend. Sec. Comput.*, vol. 14, no. 1, pp. 95–108, Jan./Feb. 2017.
[14] E. N. Alkhanak, S. P. Lee, and S. U. R. Khan, "Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities," *Future Gener. Comput. Syst.*, vol. 50, pp. 3–21, Sep. 2015.
[15] M. S. Chang, D. J. Chen, and M. S. Lin, "The distributed program reliability analysis on a star topology," *Comput. Oper. Res.*, vol. 27, no. 2, pp. 129–142, 2000.
[16] R. A. Sahner and K. S. Trivedi, "Performance and reliability analysis using directed acyclic graphs," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 10, pp. 1105–1114, Oct. 1987.
[17] E. Ramraj and A. S. Rajan, "Using multi-core processor to support network parallel image processing applications," in *Proc. Int. Conf. Signal Process. Syst.*, May 2009, pp. 232–235.
[18] X. Qiu, L. Luo, and Y. Dai, "Reliability-performance-energy joint modeling and optimization for a big data task," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Aug. 2016, pp. 334–338.
[19] D. Whitley, "A genetic algorithm tutorial," *Statist. Comput.*, vol. 4, no. 2, pp. 65–85, Jun. 1994.
[20] G. Syswerda, "Uniform crossover in genetic algorithm," *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 61–69.
[21] G. Jung, N. Gnanasambandam, and T. Mukherjee, "Synchronous parallel processing of big-data analytics services to optimize performance in federated clouds," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 811–818.
[22] H. Ke, P. Li, S. Guo, and M. Guo, "On traffic-aware partition and aggregation in mapreduce for big data applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 3, pp. 818–828, 2016.
[23] Y. Mo, L. Xing, F. Zhong, and Z. Zhang, "Reliability evaluation of network systems with dependent propagated failures using decision diagrams," *IEEE Trans. Depend. Sec. Comput.*, vol. 13, no. 6, pp. 672–683, Nov. 2016.
[24] A. Bahga and V. K. Madisetti, "Analyzing massive machine maintenance data in a computing cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 10, pp. 1831–1843, Oct. 2012.
[25] R. Tudoran, A. Costan, and G. Antoniu, "OverFlow: Multi-site aware big data management for scientific workflows on clouds," *IEEE Trans. Cloud Comput.*, vol. 4, no. 1, pp. 76–89, Jan. 2016.
[26] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: Probabilistic model checking for performance and reliability analysis," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 4, pp. 40–45, Mar. 2009.
[27] F. Machida, R. Xia, and K. S. Trivedi, "Performability modeling for RAID storage systems by Markov regenerative process," *IEEE Trans. Depend. Sec. Comput.*, vol. 15, no. 1, pp. 138–150, Jan. 2018.
[28] X. Qiu, P. Sun, X. Guo, and Y. Xiang, "Performability analysis of a cloud system," in *Proc. IEEE 34th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2015, pp. 1–6.
[29] F. Robledo and P. Sartor, "A simulation method for network performability estimation using heuristically computed pathsets and cutsets," *Int. J. Metaheuristics*, vol. 2, no. 4, pp. 370–391, 2013.
[30] X. Qiu, Y. Dai, Y. Xiang, and L. Xing, "Correlation modeling and resource optimization for cloud service with fault recovery," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 693–704, Jul./Sep. 2017.
[31] R. Urgaonkar, U. C. Kozat, K. Igarashi, and M. J. Neely, "Dynamic resource allocation and power management in virtualized data centers," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2010, pp. 479–486.
[32] P. Sun, Y. Dai, and X. Qiu, "Optimal scheduling and management on correlating reliability, performance, and energy consumption for multiagent cloud systems," *IEEE Trans. Rel.*, vol. 66, no. 2, pp. 547–558, Jun. 2017.
[33] Y. Zhao, K. Chen, W. Bai, M. Yu, C. Tian, Y. Geng, Y. Zhang, D. Li, and S. Wang, "Rapier: Integrating routing and scheduling for coflow-aware data center networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 424–432.
[34] C. Q. Wu, X. Lin, D. Yu, W. Xu, and L. Li, "End-to-end delay minimization for scientific workflows in clouds under budget constraint," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 169–181, Apr./Jun. 2015.
[35] Y.-C. Liang and A. E. Smith, "An ant colony optimization algorithm for the redundancy allocation problem (RAP)," *IEEE Trans. Rel.*, vol. 53, no. 3, pp. 417–423, Sep. 2004.
[36] T.-C. Chen and P.-S. You, "Immune algorithms-based approach for redundant reliability problems with multiple component choices," *Comput. Ind.*, vol. 56, no. 2, pp. 195–205, Feb. 2005.
[37] D. W. Coit and A. E. Smith, "Reliability optimization of series-parallel systems using a genetic algorithm," *IEEE Trans. Rel.*, vol. 45, no. 2, pp. 254–260, Jun. 1996.
[38] N. Beji, B. Jarboui, M. Eddaly, and H. Chabchoub, "A hybrid particle swarm optimization algorithm for the redundancy allocation problem," *J. Comput. Sci.*, vol. 1, no. 3, pp. 159–167, Aug. 2010.

MIN TAO received the B.S. degree in software engineering and the M.S. degree in information and communication engineering from the University of Electronic Science and Technology of China (UESTC), where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering. He is also a Researcher with the Next Generation Internet and Data Processing Technology of National Local Joint Engineering Laboratory, UESTC. His current research interests include cloud computing, performance modeling, and optimization.

XIWEI QIU received the B.S. degree in electronic and information engineering from Jilin University, and the M.S. degree in software engineering and the Ph.D. degree in computer science from UESTC.

He is currently an Assistant Professor with the Department of Computer Science, UESTC. He has published more than 20 articles, including several SCI journal articles published in IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, and IEEE TRANSACTIONS ON RELIABILITY. His current research interests include reliability modeling and analysis of complex systems and networks. He was a recipient of the Science and Technology Progress Award in Henan province, in 2019. He is also the Program Chair of the 6th International Symposium on System and Software Reliability. He also serves as a Reviewer of several SCI journals, including *European Journal of Operational Research* and *Reliability Engineering & System Safety*.

PENG SUN received the Ph.D. degree from UESTC. He is currently a Faculty Member with the Next Generation Internet and Data Processing Technology of National Local Joint Engineering Laboratory, UESTC. In the recent three years, he participated in or studied two state-level key laboratory construction projects, and published three articles in SCI journals and five articles in international high-level academic conferences. His research interests include cloud computing, reliability modeling, and optimization.