Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 7, July 2018, pp. 1487-1492
DOI: 10.23940/ijpe.18.07.p12.14871492

Reliability Evaluation of a Parallel Job with Real-Time Redundant Computing

Xiwei Qiu, Liang Luo*, Sa Meng, and Han Xu

University of Electronic Science and Technology of China, Chengdu, 611731, China

Abstract

In network systems, a job with a large amount of work-requirement is usually divided into multiple tasks to achieving parallel computing. However, if any task is failed due to random task failures or server failures, the entire job cannot be complete. In traditional redundant computing, copies of tasks are initiated whenever the tasks are found to be failed. This is not an efficient approach from the perspective of the performance. Real-time parallel computing is more high-efficient, which makes tasks and their copies run simultaneously. In this paper, to evaluating the reliability of such a job, we first describe a complicated parallel and redundant computing environment as multiple minimal-job spanning trees (MJST) consisting of tasks and servers. Then, we design two operators of MJSTs to systemically analyze complicated failure correlations among multiple MJSTs. Finally, an algorithm based on the Bayesian theorem is presented to evaluate the reliability of a parallel job with real-time parallel computing. Illustrative examples are provided.

Keywords: parallel computing; real-time redundant computing; failure correlation; reliability

(Submitted on April 14, 2018; Revised on May 23, 2018; Accepted on June 22, 2018)

1. Introduction

Network systems (e.g. cloud computing systems) have become increasingly important for integrating various computing resources via the Internet. In the network system, there exists an important kind of job that has high computational complexities, e.g., big data processing. Such a job is usually divided into multiple tasks to be executed on heterogeneous servers in parallel. However, parallel computing also brings a negative effect on reliability. That is, if any task is failed due to random task failures or server failures, the entire job cannot be complete. Therefore, redundant computing can be adopted to achieve fault tolerant. However, in traditional redundant computing, copies of tasks are initiated whenever the tasks are found to be failed. This is not an efficient approach from the perspective of the performance. Due to the network system having a large amount of resources, it can realize more high-efficient redundant computing, i.e., tasks and their copies run simultaneously [1].

Random task failures, server failures, and complicated failure correlations (i.e., failures of a server inevitably result in the tasks hosted on it to be failed simultaneously) make the reliability of such a redundant parallel job hard to be evaluated by adopting some existing reliability models [2,3,4]. For example, although our prior research [5] proposed a reliability model to investigate failure correlation between co-located tasks and the host server, it cannot be directly applied or extended to the scenario described in this paper since complicated network structure and redundant parallel computing are not taken into account.

Instead, this paper systemically studies a reliability model for a parallel job with real-time redundant computing. The primary innovation of the model is that it simplifies the complicated network structure to a set of multiple MJSTs. Then, we propose an extended expression pattern that expresses a MJST to a set of two-filed records. Then, two important operators of MJSTs are designed to analyze failure correlations among multiple MJSTs systemically. Finally, an evaluation algorithm

* Corresponding author.
E-mail address: 376454384@qq.com

based on the Bayesian theorem is presented to obtain the probability distribution of random job completion time, which can be used to calculate the reliability of the redundant parallel job.

2. Real-Time Redundant Computing in the Star-Structured Network

In principle, the network system has an operating system to perform the centralized management of a large amount of heterogeneous resources. The real-time redundant computing of a parallel job can be realized by the operating system with the following steps: 1) the operating system receives a parallel job, and becomes the start and terminal point to execute the job; 2) it divides the job into multiple tasks and creates copies for the tasks (since each copy is essential a task, in the following content, we also use term 'task' which sometimes means a copy); 3) all the tasks are sent to heterogeneous servers to be executed simultaneously. Fig. 1 shows a scenario where a parallel job is redundantly executed in the star-structured network.

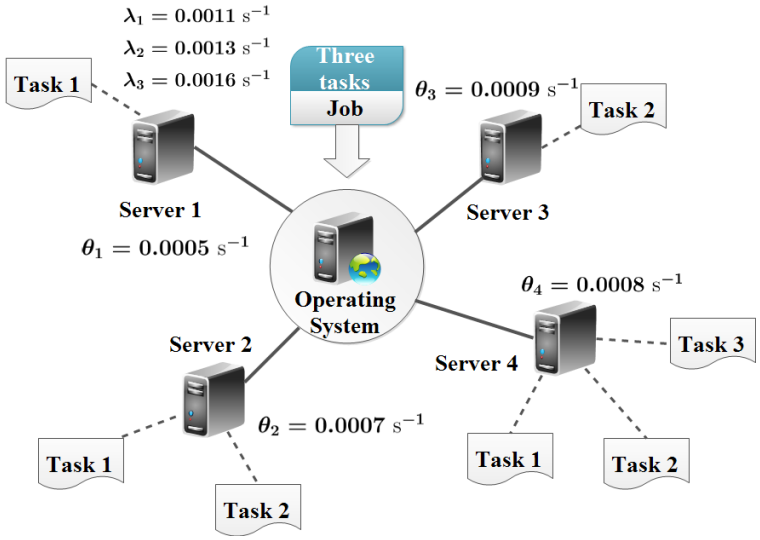

Figure 1. A scenario that a parallel job is redundantly executed in the star-structured network

Suppose there are $N\ (N \geq 1)$ servers that are assigned to run the job, and the job is divided into $M\ (M \geq 1)$ tasks. As shown in Figure 1, $N = 4$ and $M = 3$. Parameters θ_n and λ_m are the failure rates of server n and task m, respectively. The entire job has work requirement w, which can be quantified as the number of instructions that need to be processed. Since the job is divided into M tasks in total, the work requirement of task m is w_m, and equation $w = \sum_{m=1}^{M} w_m$ is satisfied. Failure times of server n and task m follow exponential distributions with failure rates of θ_n and λ_m, respectively. Failures of different servers are assumed to be independent. Meanwhile, consider that recent isolation techniques (particularly, the cloud virtualization technique) are widely adopted in the network system, failures of different tasks including co-located tasks are also assumed to be independent.

3. Reliability Model

If task $m\ (m = 1, 2, \ldots, M)$ is assigned to server n, its computational speed can be obtained as $c_{mn} = f_n \cdot u_{mn}$, where f_n and u_{mn} are the maximal CPU frequency of the server and the percentage of CPU resources occupied by the task. Let t_{mn} denote the fault-free execution time of task m, it can be derived by $t_{mn} = w_m / c_{mn}$.

3.1. Extended Expression Pattern of a MJST

A *minimal job spanning tree* (MJST) is a minimal possible combination of necessary elements that guarantee the execution of the entire job. The elements in a MJST include M different tasks and their host servers. Given a concrete job execution strategy expressed in the form of $(\phi_1, \phi_2, \ldots, \phi_M)$, the total number of all possible MJSTs is $K = \prod_{m=1}^{M} |\phi_m|$. For example, as shown in Fig. 1, the job execution strategy is ($\phi_1 = \{1, 2, 4\}$, $\phi_2 = \{2, 3, 4\}$, $\phi_3 = \{4\}$), and $K = 9$.

Let S_k represent the element set of MJST k $(k = 1, 2, \ldots, K)$. The extended expression pattern of MJST k is given as

$$S_k = \left\{ \underbrace{\big(e_1, t(e_1)\big), \ldots, \big(e_M, t(e_M)\big)}_{tasks}, \underbrace{\big(e_{M+1}, t(e_{M+1})\big), \ldots, \big(e_{D_k}, t(e_{D_k})\big)}_{SNs} \right\}, \quad D_k = |S_k| \tag{1}$$

where $(e_d, t(e_d))$ is the dth element in S_k expressed as a two-filed record. e_d and $t(e_d)$ are the identifier and the fault-free execution time of the element. As seen in (1), the former M elements are M different tasks, and the rest elements are servers occupied by running those tasks. If a server hosts multiple tasks, its fault-free execution time is the maximal value of the fault-free execution times of the co-located tasks. Suppose t_k is the fault-free execution time of S_k, it can be derived as $t_k = \max_{d=1,\ldots,D_k}\{t(e_d)\}$, where $D_k = |S_k|$. Let $\Pr(S_k)$ represents the probability that all elements in set S_k do not fail during their fault-free execution times. From (1), it can be obtained by

$$\Pr(S_k) = \prod_{d=1}^{M} e^{-\lambda_d \cdot t(e_d)} \cdot \prod_{d=M+1}^{D_k} e^{-\theta_d \cdot t(e_d)} \tag{2}$$

3.2. Two Operators of MJSTs to Analyze Failure Correlations Among Multiple MJSTs

Let F_k denote the events that MJST S_k is not failed during time t_k, that is, $\Pr(F_k) = \Pr(S_k)$. We design two operators to find critical element sets $S_{j|i}, S_{jk|i}, S_{jkl|i}, \ldots$ that result in event $F_j|F_i, F_jF_k|F_i, F_jF_kF_l|F_i, \ldots$, respectively. The corresponding operations of MJSTs are given as follows.

- **Operation 1**: $S_{j|i} = S_j - S_i$
  Definition: S_j and S_i are the element sets of MJSTs j and i, respectively. Operator '$-$' is used to found the critical elements that do not affect the run of MJST i, but decides if MJST j is complete. $S_{j|i} = S_j - S_i$ is realized by: for all $\big(e_d, t(e_d)\big) \in S_j$, 1) if element identifier e_d cannot be found in S_i, then let $\big(e_d, t(e_d)\big) \in S_{j|i}$; 2) else get the fault-free execution time of e_d in S_i, and suppose it is $t'(e_d)$. If $t(e_d) > t'(e_d)$, then let $\big(e_d, t(e_d) - t'(e_d)\big) \in S_{j|i}$.

- **Operation 2**: $S_{jk|i} = S_{j|i} + S_{k|i}$
  Definition: $S_{j|i}$ and $S_{k|i}$ are two element sets that do not affect the run of MJST i. Operator '+' is used to found the critical elements that decide if the event $F_jF_k|F_i$ happens. $S_{jk|i} = S_{j|i} + S_{k|i}$ is realized by: 1) for all $\big(e_d, t(e_d)\big) \in S_{j|i}$, if element identifier e_d cannot be found in $S_{k|i}$, then let $\big(e_d, t(e_d)\big) \in S_{jk|i}$; else get the fault-free execution time of e_d in $S_{k|i}$ (suppose it is $t'(e_d)$). Let $\big(e_d, \max[t(e_d), t'(e_d)]\big) \in S_{jk|i}$ and delete $\big(e_d, t'(e_d)\big)$ from $S_{k|i}$. 2) For all $\big(e_d, t(e_d)\big) \in S_{k|i}$, let $\big(e_d, t(e_d)\big) \in S_{jk|i}$.

Use operations 1 and 2, we can find all critical element sets that result in events $F_j|F_i, F_jF_k|F_i, F_jF_kF_l|F_i, \ldots$. For example, element set $S_{jkl|i}$ resulting in event $F_jF_kF_l|F_i$ can be obtained by first getting $S_{j|i} = S_j - S_i$, $S_{k|i} = S_k - S_i$, $S_{l|i} = S_l - S_i$ according to operation 1. Then, $S_{jkl|i} = S_{j|i} + S_{k|i} + S_{l|i}$ can be derived by using operation 2 iteratively.

3.3. Reliability Evaluation based on the Bayesian Theorem

After finding all possible MJSTs (i.e., S_1, S_2, …, S_K), we present an algorithm to evaluate the reliability of the parallel job with real-time redundant computing. The algorithm is based on the Bayesian theorem, which is described as follows.

- Step 1: Sort element sets S_1, S_2, …, S_K in an increasing order of their fault-free execution times (i.e., t_k), and make element sets that have an identical fault-free execution time constitute a group. Suppose there are Z groups in total $(1 \le Z \le K)$, and x_z is the fault-free execution time of group z $(z = 1, 2, \ldots, Z)$. After grouping S_1, S_2, …, S_K, the fault-free execution time of the groups are strictly increasing in z, i.e., $0 < x_1 < \cdots < x_z < \cdots < x_Z$.

- Step 2: Suppose T is random job completion time. For the minimal value of T (i.e., $T = x_1$), we can first derive $p(x_1) = \Pr(T = x_1) = \Pr(E_1)$, where E_z is the event that at least one of the MJSTs from group z finishes the job. Suppose there are K_z MJSTs in group z $(\sum_{z=1}^{Z} K_z = K)$. Event E_z can be expressed as $E_z = F_1 \cup F_2 \cup \cdots \cup F_{K_z}$. Now, using the Bayesian theorem on conditional probability, $p(x_1)$ can be written as

$$p(x_1) = \Pr(F_1) + \sum_{i=2}^{K_1} \Pr(F_i) \cdot \Pr(\overline{F}_{i-1} \cdots \overline{F}_1 | F_i) \tag{3}$$

- Step 3: The other probability for $T = x_z$ $(z = 2, 3, \ldots, Z)$ can be calculated by $p(x_z) = \Pr(E_z \overline{E}_{z-1} \overline{E}_{z-1} \cdots \overline{E}_1)$, where $\overline{E}_z$ is the event that none of the MJSTs from group z finishes the job. Note that $\overline{E}_z$ can be expressed as $\overline{E}_z = \overline{F}_1 \cdots \overline{F}_i \cdots \overline{F}_{K_z}$. Therefore, $p(x_z)$ can be calculated by

$$p(x_z) = \sum_{i=1}^{K_z} \Pr(F_i) \cdot \Pr(\overline{F}_{i-1} \cdots \overline{F}_1 \overline{E}_{z-1} \cdots \overline{E}_1 | F_i) \tag{4}$$

- Step 4: As seen in (3) and (4), there exists a form of probabilities that is hard to be calculated directly, i.e., $\Pr(\overline{F}_s \cdots \overline{F}_1 | F_i)$. For calculating probabilities having such a form, we can use

$$\begin{aligned} \Pr(\overline{F}_s \cdots \overline{F}_1 | F_i) &= 1 - \Pr(F_s \bigcup F_{s-1} \cdots \bigcup F_1 | F_i) \\ &= 1 - \sum_{j=1}^{s} \Pr(F_j | F_i) + \sum_{j,k}^{j \neq k} \Pr(F_j F_k | F_i) - \sum_{j,k,l}^{j \neq k \neq l} \Pr(F_j F_k F_l | F_i) + \cdots + (-1)^s \Pr(F_s \cdots F_1 | F_i) \end{aligned} \tag{5}$$

  In (5), the critical element sets resulting in all the events (i.e., $F_j|F_i, F_jF_k|F_i, F_jF_kF_l|F_i, \ldots, F_s \cdots F_1|F_i$) can be obtained by using operations 1 and 2. Having the critical element sets, all probabilities in (5) can be calculated from (2). Now, substitute (5) into (3) and (4), $p(x_1)$ and $p(x_z)$ can be calculated, respectively.

- Step 5: The reliability of the parallel job with real-time redundant computing is defined as the probability of job completion. Denoted it as R, which be derived as

$$R = \sum_{z=1}^{Z} p(x_z) \tag{6}$$

4. Numerical Examples

To begin the experiments, the parameters first need to be estimated. The processing speed of a SN can be measured by mapping the CPU frequency onto the MIPS rating. The reliability parameters (i.e., the failure rates of servers and tasks) can be obtained by letting the operating system accumulates two indices: the total running time (τ) and the number of failures (σ) of individual tasks and servers. Then, according to maximum likelihood estimation, the operating system can estimate the failure rates of the individuals by calculating σ/τ [6].

Take the scenario of Fig. 1 as an example. The estimated parameters of tasks and servers are listed in Table 1.

Table 1. Parameters of tasks and servers

Parameters of tasks				Parameters of servers		
No.	Failure rate (s^{-1})	Work requirement	The number of occupied core	No.	Failure rate (s^{-1})	CPU Information
1	0.0011	3.5×10^5 million	2 cores	1	0.0005	Xeon E5450 8×3.0 GHz
2	0.0013	8.0×10^5 million	4 cores	2	0.0007	Xeon L5530 8×2.4 GHz
3	0.0016	7.2×10^5 million	6 cores	3	0.0009	Xeon E5-2655 16×2.2 GHz
				4	0.0008	Xeon E5-2640 12×2.5 GHz

As shown in Figure 1, the job execution strategy is $\phi_1 = \{1, 2, 4\}$, $\phi_2 = \{2, 3, 4\}$, $\phi_3 = \{1\}$. We can find nine MJSTs in total (i.e., $K = 9$). According to the parameters given in Table 1, the fault-free execution time of all elements in each MJSTs can be calculated. Then, the extend expression patterns of all MJSTs can be derived from (1), which is listed in Table 2.

Table 2. Extended expression patterns of all MJSTs

MJST	Extended expression	Fault-free execution time (s)
S_1	{(A_{11},58), (A_{22},83), (A_{34},48), (N_1, 58), (N_2,83), (N_4,48)}	83
S_2	{(A_{11},58), (A_{23},91), (A_{34},48), (N_1, 58), (N_3,91), (N_4,48)}	91
S_3	{(A_{11},58), (A_{24},80), (A_{34},48), (N_1, 58), (N_4,48)}	80
S_4	{(A_{12},73), (A_{22},83), (A_{34},48), (N_2,83), (N_4,48)}	83
S_5	{(A_{12},73), (A_{23},91), (A_{34},48), (N_2, 73), (N_3,91), (N_4,48)}	91

S_6	{(A_{12},73), (A_{24},80), (A_{34},48), (N_2,73), (N_4,80)}	80
S_7	{(A_{14},70), (A_{22},83), (A_{34},48), (N_2,83), (N_4,70)}	83
S_8	{(A_{14},70), (A_{23},91), (A_{34},48), (N_3,91), (N_4,70)}	91
S_9	{(A_{14},70), (A_{24},80), (A_{34},48), (N_4,80)}	80

The identifiers in an extended expression have two forms: A_{mn} means task m is assigned to server n, and N_n is host server n. Having the fault-free execution times of the MJSTs, we can divide them into three groups: $G_1 = \{S_3, S_6, S_9\}$ with $x_1 = 80$, $G_2 = \{S_1, S_4, S_7\}$ with $x_2 = 83$, $G_3 = \{S_2, S_5, S_8\}$ with $x_3 = 91$. According to the algorithm described in section 3.3, we first calculate $p(x_1) = \Pr(E_1)$. From (3), it can be written as

$$\Pr(E_1) = \Pr(F_3) + \Pr(\overline{F}_3 F_6) + \Pr(\overline{F}_3 \overline{F}_6 F_9) = \Pr(F_3) + \Pr(F_6)\Pr(\overline{F}_3|F_6) + \Pr(F_9)\Pr(\overline{F}_3 \overline{F}_6|F_9) \quad (7)$$

We can use the algorithm's step 4 to calculate all probabilities in (7). For example, to calculate $\Pr(\overline{F}_3|F_6)$, we can use the operation presented in section 3.2 to find the critical element set, that is, $S_{3|6} = S_3 - S_6 = \{(A_{11}, 58), (N_1, 58)\}$. Then, $\Pr(\overline{F}_3|F_6) = 1 - \Pr(F_3|F_6) = 1 - \Pr(S_{3|6}) = 0.0886$. After calculating all probabilities in (7), $p(x_1) = \Pr(T = 80)$ is derived as 0.7822. Now, we can further calculate the other probabilities $p(x_2), \ldots, p(x_Z)$. For example,

$$p(x_2) = \Pr(E_2 \overline{E}_1) = \Pr(F_1)\Pr(\overline{E}_1|F_1) + \Pr(F_4)\Pr(\overline{F}_1 \overline{E}_1|F_4) + \Pr(F_7)\Pr(\overline{F}_4 \overline{F}_1 \overline{E}_1|F_7) \quad (8)$$

which can be calculated as 0.0961, and $p(x_3)$ can also be derived as 0.1127. Finally, the reliability of the redundant parallel job shown in figure 1 is evaluated as $R = p(x_1) + p(x_2) + p(x_3) = 0.8873$. For verifying the correction of the proposed reliability model, we also design a simulation experiment based on the Monte Carlo method, which simulates the entire execution process of the job. Figure 2 illustrates root-mean-square errors (RMSE) of simulation results derived by running the simulation program at different times. As shown in the figure, RMSE of the expected job completion time (i.e., $\sum_{z=1}^{Z} x_z p(x_z)$) decreases with the increase of the simulation times, which means statistical results are very close to the theoretical result evaluated by the proposed model (the RMSE is almost 0.0001s when the simulation times is larger than 10000). This effectively proves that our theoretical model is justified.

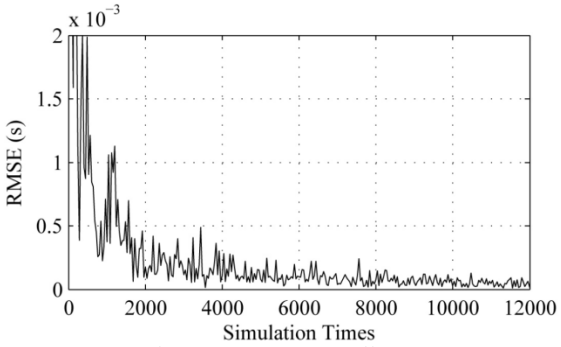

Figure 2. RMSEs of simulation results at different simulation times

5. Conclusions

In this paper, we proposed a theoretical model for evaluating the reliability of a parallel job with real-time redundant computing. Theoretical modeling is always a critical research filed that can contribute to providing important information to various applications in realistic environments. For example, it can help the operating system make an optimal job execution strategy. There are many studies focused on building the theoretical model for analyzing the reliability metric. For example, Jung et al. applied a theoretical model emphasizing network delay for a parallelized task [7]. Ke et al. also proposed a theoretical model considering network topology, network traffic, and data size for a parallelized task processed by the MapReduce mechanism [8]. Bahaga and Madisetti presented a reliability model capturing numerous failures of a Hadoop system with a large number of machines [9]. However, those models do not consider real-time redundant computing, and cannot be directly applied or extended to the scenario described in this paper.

In fact, reliability evaluation is not trivial but important, since it has a significant effect on other metrics, such as performance and energy consumption. Our prior work proposed a flexible modeling approach based on the Bayesian theorem for evaluating performability metric, which depicts random performance is seriously affected by reliability-performance correlation [10]. Our prior work has also investigated reliability-energy correlation for a big data task, which

shows that reliability factors also make the energy consumption metric become a complicated random variable [11]. But those works do not consider the important common caused failure of co-located tasks caused by failures of the host server. The reliability model presented in this paper can effectively fix this lack, and can further connect with performance models or energy consumption models for building a more comprehensive reliability-performance-energy correlation model.

Due to various application environments, redundant parallel jobs in the network system may become more complicated. For example, precedence constraints on tasks can make failures of a task possibly affect other tasks. Extending the proposed reliability model to more complicated application environments will be studied in our future work.

Acknowledgements

This work is supported by National Natural Science Foundation of China (No. 61602094).

References

1. Y. Jiang, "A Survey of Task Allocation and Load Balancing in Distributed Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 585-599, 2015
2. J. Cha, M. Guida, and G. A. Pulcini, "Competing Risks Model With Degradation Phenomena and Catastrophic Failures," *International Journal of Performability Engineering*, vol. 10, no. 1, pp. 63-74, 2014
3. Y. Dai, Y. Pan, and X. Zou, "A Hierarchical Modeling and Analysis for Grid Service Reliability," *IEEE Transactions on Computer*, vol. 56, no. 5, pp. 681-691, 2007
4. X. Ni, J. Zhao, W. Song, and H. Li, "Reliability Modeling for Two-stage Degraded System Based on Cumulative Damage Model," *International Journal of Performability Engineering*, vol. 12, no. 1, pp. 89-94, 2016
5. 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 Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 3, pp. 401-412, 2016
6. M. Xie, Y. Dai and K. L. Poh, "Computing Systems Reliability: Models and Analysis," Kluwer, New York, 2004.
7. G. Jung, N. Gnanasambandam, and T. Mukherjee, "Synchronous Parallel Processing of Big-data Analytics Services to Optimize Performance in Federated Clouds," in *Proceedings of 2012 IEEE 5th International Conference on Cloud Computing*, pp. 811-818, 2012
8. H. Ke, P. Li, S. Guo, and M. Guo, "On Traffic-aware Partition and Aggregation in MapReduce for Big Data Applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 3, pp. 818-828, 2015
9. A. Bahga, and K. Madisetti, "Analyzing Massive Machine Maintenance Data in a Computing Cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 10, pp. 1831-1843, 2011.
10. X. Qiu, P. Sun, X. Guo, and Y. Xiang, "Performability Analysis of a Cloud System," in *proceedings of 2015 IEEE 34th International Performance Computing and Communications Conference*, pp. 1-6, 2016
11. X. Qiu, L. Luo, and Y. Dai, "Reliability-performance-energy Joint Modelling and Optimization for a Big Data Task," in *Proceedings of 2016 IEEE International Conference on Software Quality, Reliability and Security Companion*, pp. 334-338, 2016