Modeling and analyzing cost-aware fault tolerant strategy for cloud application

Liqiong Chen[1,2], Guisheng Fan[3], Yunxiang Liu[1]
[1]Department of Computer Science and Information Engineering
Shanghai Institute of Technology, Shanghai 200235, China
[2]Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, China
[3] Department of Computer Science and Engineering
East China University of Science and Technology, Shanghai 200237, China
Correspondence should be addressed to Guisheng Fan; gsfan@ecust.edu.cn

[1] ***Abstract*—In this paper, we propose a method to model and analyze cost-aware fault tolerant strategy for cloud computing. First, Petri nets are used to describe the structure of cloud computing, including component, cloud service and cloud application, thus forming the fault tolerant model of cloud computing. Second, a dynamic fault tolerant strategy is proposed, which can dynamically make fault tolerant strategy with the lowest cost based on the current state and failed component. Third, we present operational semantics and related theories of Petri nets for establishing the correctness of our proposed method. We have also performed a series of simulations to evaluate our proposed approach. Results show that it can help reveal the structural and behavioral characteristics of cloud computing, and reduce the fault tolerant cost.**

***Index Terms*—Cloud computing, fault tolerance, cost, Petri net, reliability**

I. Introduction

Cloud computing sets a new paradigm for infrastructure management by offering unprecedented possibilities to deploy software in distributed environments[1]. While fault tolerant processing techniques are mainly used for the development of reliable distributed systems [2]. However, cloud computing may include a number of cloud applications. The fault tolerant strategy of cloud application may affect each other, and different strategies may have different costs. It is a challenging task to dynamically make the fault tolerance strategy with the lowest cost for the failed cloud components. In addition, most of the previous methods on fault tolerant process for cloud application didn't consider the user's QoS constraints (such as time, cost, etc.). If the cost of cloud application is high, then the users will be unwilling to use cloud service, which in turn would cause the loss of users' interest in the cloud service.

This paper investigates how to model and analyze cost aware fault tolerant strategy for cloud computing. Below summarizes our main contributions: First, we provide a flexible way for designers to specify their requirements, Petri nets are used to model different components of cloud computing. Second, we propose a method to dynamically make fault tolerant strategy, which can get the fault tolerant strategy with the lowest cost based on the current state and failed component. The process that many cloud applications compete for cloud service is converted into the optimization of fault tolerant strategy by considering the recovery cost. Third, the operational semantics and related theories of Petri nets help establish the effectiveness of our proposed method.

[1]DOI reference number: 10.18293/SEKE2016-247

The remainder of this paper is organized as follows. Section II describes how we model the different components of cloud application. Next, Section III proposes the fault tolerant strategy and analysis technique, and then evaluate the proposed method via a series of simulations (Section IV). Section V surveys related work, and Section VI concludes.

II. Modeling cloud application

A. Requirements of cloud computing

Because cloud computing may include several cloud applications. As the function of cloud application is composed by a number of independent sub-functions (component) according to a certain composition rules [3], each component has several cloud services which can realize its function.

Definition 1: The fault tolerant requirement of cloud application is 6-tuple: $\Xi = (C, WS, A, RC, RW, RL)$: (1) C is the finite set of component in cloud application. (2) $WS = (WS^c, WS^e)$ is the finite set of service, WS^c, WS^e are the set of matching service and replacement service. (3) A is the finite set of cloud application. (4) $RC : C \rightarrow (0, 1)$ is the weight of component. (5) $RW = (CC, CE)$ is the attribute function of service, CC is the reliability of service. $CE(WS_i^e) = (ae_i, se_i, at_i, st_i)$ is the cost, starting cost, running time and start-up time of replacement service. (6) $RL = (RLr, RLe, RLc)$, RLr is the relation function between the components. RLe, RLc are the set of replacement service and matching service of component.

The fault tolerant strategy of cloud application is composed by a series of replacement services. SP = (SP_1, ..., SP_f) is the fault tolerant strategy of cloud applications a_1, ... a_f. SP_i = $\{WS_k, \ldots, WS_g\}$ describes the replacement service of all components in a_i.

B. Syntax and semantics

Petri net (*PN*) is a formal language for describing the distributed system because its semantics is formally defined[4].

Definition 2: A 6-tuple $\Sigma = (PN, IO, tr, D, A_F, M_0)$ is called a Fault tolerant Model(FTM): (1) $PN = (P, T, F, W)$ is a basic Petri net. P, T, F, W are the finite set of place, transition, arc and weight. (2) IO is a special type of place, which is the interface of Σ and denoted by dotted circle. (3) tr is the attribute function of transition, $tr(t_i)$ = (λ_i, pi_i, r_i, ct_i) is the firing probability(deterministic), cost, priority and running time of transition, the default value is (1, 0, 0, 0) The smaller the value of r_i, the higher the priority of transition, the priority of instantaneous transition is higher than time transition. (4)

D = $\{d^c_{i,j}, d^w_k, \varphi\}$ is the individuality of component, the matching service and data packet φ. (5) A_F is the formula set or individuality on the arc. (6) M_0 is the initial marking of FTM.

The input/output arc of transition t_i and the free variable in $A_T(t_i)$ are denoted by $FV(t_i)$=$\{x_1, x_2, \ldots, x_n\}$. S=(M, EC, ET) is the state of model, EC and ET are the cost and time when the system reaches S, the initial state S_0=(M_0, EC_0, ET_0), EC_0=ET_0=0. $\forall t \in T$, if $FV(t_i)$=$\{x_1, x_2, \ldots, x_n\}$, the token $\{d_1, d_2, \ldots, d_n\}$ meets $d_i \in \{M(p) \mid p \in^\bullet t \cup t^\bullet\}$ and d_i corresponds to the variable x_i, the instance of t is obtained by replacing $d_1, d_2, \ldots, d_n$ with $x_1, x_2, \ldots, x_n$, which is called a replacement of t, denoted by $t < x_1 \leftarrow d_1, x_2 \leftarrow d_2, \ldots, x_n \leftarrow d_n >$, it can be denoted by $t < d_1, d_2, \ldots, d_n >$. The replacement is mainly used to bind the token to the input/output arc of t and all free variables in $A_T(t)$. Let $A_T(t) < d_1, d_2, \ldots, d_n >$ and $A_F(p,t) < d_1, d_2, \ldots, d_n >$ be the values got by replacing $A_T(t)$ and $A_F(p,t)$ of input arc with $d_1, d_2, \ldots, d_n$.

Definition 3: If $t < d_1, d_2, \ldots, d_n >$ makes $A_T(t) < d_1, d_2, \ldots, d_n > \wedge A_F(t) < d_1, d_2, \ldots, d_n >$=true, then $t < d_1, d_2, \ldots, d_n >$ is called the feasible replacement of transition t under state S.

All the feasible replacements of transition t under S are denoted by set $VP(S,t)$. If $VP(S,t) \neq \emptyset$, then t is enable under S, denoted by $S[t >$. Let $ET(S) = \{t|S[t >\}$. For $t_i \in ET(S)$, if $\lambda_i \leq min(\lambda_j)$, $t_j \in ET(S)$, then the firing of t_i under S is effective. All effective firing transitions under state S are denoted by $FT(S)$. The process that S reaches S' by firing a feasible replacement $t_i < d_1, d_2, \ldots, d_n >$ of t_i is denoted by $S[t_i < d_1, d_2, \ldots, d_n > S'$. If the relation between transitions is parallel, then the firing of any transition cannot affect the firing of another transition. The concurrent transitions under S are denoted by set $MT(S)$.

Definition 4: $H(S)$=$\{t < d_1, d_2, \ldots, d_n > | t \in MT(S), t < d_1, d_2, \ldots, d_n > \in VP(S,t) \wedge (ct_i \leq min(ct_f), t_f \in MT(S')) \wedge \lambda_i = \infty\}$ is called the greatest firing set of S.

FTM can reach a new state S' by firing the greatest firing set $H(S)$ under the state S.

Definition 5: Let Ω be a *FTM* model, S is a state of Ω, the system can reach a new state S' by effectively firing $H(M)$, denoted by $S[H(S) > S'$, then S' is called a reachable state of S. S' is computed based on the following rules:

(1) $\forall t_i < d_1, d_2, \ldots, d_n > \in H(S)$, $\forall p_j \in^\bullet t_i \cup t_i^\bullet$: $M'(p_j) = M(p_j) - A_F(p_j, t_i) < d_1, d_2, \ldots, d_n > + A_F(t_i, p_j) < d_1, d_2, \ldots, d_n >$.

(2) Computing EC': EC'= EC+ $\sum_{t_i \in H(S)} \pi_i$.

(3) Computing ET': ET'=ET+max$\{ct_i\}$, $t_f \in H(S)$.

All the possibly reachable states of S are denoted by $R(S)$. *FTM* will start from the initial state S_0 and generate the new state by effectively firing the enabled transitions. $\delta(S_i, S_j)$ is the firing sequence from S_i to S_j.

C. Modeling basic elements

Modeling component. The *FTM* model of component $C_{i,j}$ is shown in Fig.1, where D = $\{d^c_{i,j}, d^w_k, \varphi\}$ is the individuality of component, which represents the component $C_{i,j}$, the matching service WS_k and data packet φ. The initial resource distribution $M_0(p_{ws,i,j})$ = d^w_k. A_F is the formula set or individuality on the arc, such as $A_f(p_{ws,i,j}, t_{s,i,j}) = x$.

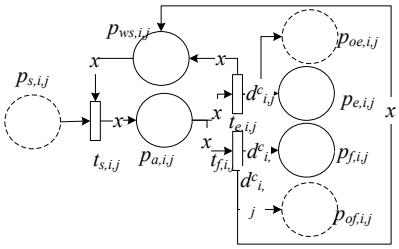

Fig. 1. Modeling component

Modeling replacement service. The *FTM* model of WS_i is shown in Fig.2, tr is the attribute function of transition. so the firing time of $t_{e,i}$ is equal to the execution time of service. The initial distribution of resources is $M_0(p_{I,i})$ = d^w_i.

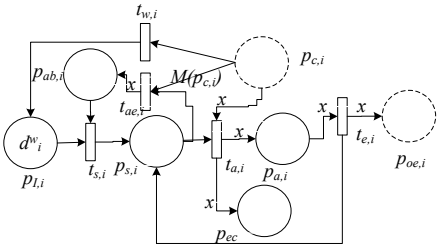

Fig. 2. Modeling replacement service

Modeling cloud application. The steps for constructing cloud application are as follows. 1) Matching the resources for each component based on the requirement, then construct the model of all components. 2) Introducing $t_{s,i}$ and $p_{s,i}$ to describe the beginning operation and position of the application, then initialize the system based on the characteristics of cloud application. At the same time, we introduce $t_{e,i}$ and $p_{e,i}$ to describe the termination operation and position of the system. 3) Composing the model of each element based on the basic relationships between each element. 4) Introducing $p_{af,i}$ to store all failed component in a_i. 5) Setting $M_0(p_{s,i}) = \varphi$.

Modeling fault tolerant processing. $t_{af,i}$ is used to upload all failed components in cloud application a_i to the place p_{fc}. p_{ench} is used to store the fault tolerant strategy of each cloud application. If any component fails ($|p_{fc}| \neq 0$), then fire t_{ench} to allocate the component to the replacement service according to the fault tolerant strategy. $t_{oe,i}$ is used to transfer the results of replacement service WS_i to the cloud application.

Modeling cloud computing $\Omega(sp^i_1, sp^k_2, \ldots, sp^f_l)$***.*** The steps for constructing cloud computing are as follows. 1) Constructing the fault tolerant model of component, replacement service and cloud application. 2) Modeling the adopted fault tolerant strategy for cloud computing, sp^i_j represents that a_j takes the i-th fault tolerant strategy, sp^i_j = $\{WS_k, \ldots, WS_m\}$. 3) Merging the same place and transition, then set the initial marking.

III. Fault tolerant strategy

A. Fault tolerant cost

Let the current fault tolerant strategy be sp_d. And the configured replacement service of $C_{i,j}$ under sp_d is WS_f, which is denoted by $exC(C_{i,j}, sp_d) = WS_f$. The principle of fault tolerant strategy is to select the replacement service for failed component, so $WS_f \in RW(C_{i,j})$ and WS_f is unique.

$exW(WS_f, sp_d)$=$\{C_{i,j}|C_{i,j} \in C \wedge exC(C_{i,j}, sp_d) = WS_f\}$ is called the finished component set of WS_f under sp_d.

The recovery time of component $C_{i,j}$:

$$Alt(C_{i,j}, sp_d)=\begin{cases} at_f + st_f, exW(WS_f, sp_d) = C_{i,j}: \\ \frac{st_f + at_f \times |exW(WS_f, sp_d)|}{|exW(WS_f, sp_d)|}: \quad else \end{cases}$$

In the same way, the recovery cost of component $C_{i,j}$:

$$Ale(C_{i,j}, sp_d)=\begin{cases} se_f + ae_f, exW(WS_f, sp_d) = C_{i,j}: \\ \frac{se_f + ae_f \times |exW(WS_f, sp_d)|}{|exW(WS_f, sp_d)|}: \quad else \end{cases}$$

The above formula computes the recovery cost and time of component from the view of the reliability of matching service. The recovery of component must consider the time and cost. We will give the recovery cost of component in the following:

The fault tolerant cost of $C_{i,j}$ under sp_d is: $Al(C_{i,j}, sp_d)= x_1 \times \frac{max\_Alt(C_{i,j}) - Alt(C_{i,j})}{max\_Alt(C_{i,j}) - min\_Alt(C_{i,j})} + X_2 \times \frac{max\_Ale(C_{i,j}) - Ale(C_{i,j})}{max\_Ale(C_{i,j}) - min\_Ale(C_{i,j})}$, where $x_1 + x_2 = 1$

The fault tolerant cost of a_i under sp_d is equal to the fault tolerant cost of failed components:

$$Al(a_i, sp_d)= \prod_{C_{i,j} \in C_i} RW(C_{i,j}) \times Al(C_{i,j}, sp_d)$$

The fault tolerant cost of cloud application under the strategy sp_d is: $Al(sp_d)=\prod_{a_i} Al(a_i, sp_d)$

B. Dynamic fault tolerant strategy

Let the initial value of current fault tolerant strategy sp_d be $\{\emptyset, \emptyset, \ldots, \emptyset\}$. We will make dynamic fault tolerant strategy based on the following steps when component $C_{i,j}$ fails.

(1) $\forall WS_f \in RW(C_{I,j})$, let sp_{temp}=(sp_1^t, sp_2^t, …, sp_{i-1}^t, $sp_{i,f}^{t'}$,…, sp_k^t), that is, the system will select service WS_f for $C_{i,j}$ to recovery under sp_i^t, then compute $Al(C_{i,j}, sp_{i,f}^{t'})$.

(2) The system will select WS_f with the minimum value of $Al(C_{i,j}, sp_{i,f}^{t'})$ as the replacement service of $C_{i,j}$: sp_d = (sp_1^d, sp_2^d, …, sp_{i-1}^d, $sp_i^{'}$, …, sp_k^d), where $sp_i^{'} = sp_i^d \cup \{(C_{i,j}, WS_f)\}$, and sp^d will be viewed as the current strategy.

According to the above steps, we can ensure that cloud computing can dynamically consider the fault tolerant cost. We can weave the dynamic fault tolerant strategy into the transitions of fault tolerant model.

Definition 6: Let Ω be a fault tolerant model, S is a state of Ω, $H(S)$=$\{t < d_1, d_2, \ldots, d_n > |t \in MT(S), t < d_1, d_2, \ldots, d_n > \in VP(S,t)\}$ be the greatest concurrent set of S, $M(p_{ench})$=d_d^s, d_d^s is the token of current strategy. We can further set $H(S)$ according to the dynamic fault tolerant strategy: If $\forall t_{f,i,j} \in FT(S)$, $Al(C_{i,j}, sp_{i,f}^{t'})$=$min\{Al(C_{i,j}, sp_{i,k}^{t'})\}$, d_k^w, $d_f^w \in \#2(d_{i,j}^c)$, then $F(t_{ench}, p_{c,f}) < x \leftarrow d_{i,j}^c >$

We will analyze the correctness of dynamic fault tolerant strategy based on the internal mechanism of cloud computing.

Theorem 1: Let Ω be the fault tolerant model of cloud application, $R(\Omega)$ be the reachable state set which is obtained by using fault tolerant strategy. $\forall \in R(\Omega)$, $EW(S)$ is the firing set of cloud service when the system reaches S, $\forall WS_i \in EW(S)$, WS_i is the replacement service of $C_{f,k}$ then: $\forall WS_j \in WS - EW(S)$, if $C_{f,k} \in RC(WS_i) \cap RC(WS_j)$, then $Al(C_{f,k}, sp_{f,i}^{t'}) \leq Al(C_{f,k}, sp_{f,j}^{t'})$.

Proof by contradiction: $\exists WS_i \in EW(S)$, $\exists WS_j \in WS - EW(S)$, $C_{f,k} \in RC(WS_i) \cap RC(WS_j)$, then $Al(C_{f,k}, sp_{f,i}^{t'}) > Al(C_{f,k}, sp_{f,j}^{t'})$. Because $WS_i \in EW(S)$, S_1 is in one of the firing sequence from S_0 to S, which makes $t_{ench} \in FT(S_1)$. Because $C_{f,k} \in RC(WS_i) \cap RC(WS_j)$, therefore $d_{f,k}^c \in S(p_{fc})$. According to the definition of feasible replacement, we can get $t_{ench} < x \leftarrow d_{f,k}^c >$ and $t_{ench} < y \leftarrow d_{f,k}^c >$ are two feasible replacements of t_{ench}. Because $Al(C_{f,k}, sp_{f,i}^{t'}) > Al(C_{f,k}, sp_{f,j}^{t'})$, according to the Definition 3, we can get $H(S_1) = H(S_1) \wedge t_{ench} < x \leftarrow d_{f,k}^c >$, therefore, $\exists WS_j \in EW(S)$, which is contradicted with $WS_j \in WS - EW(S)$, the assumption does not establish.

Theorem 1 illustrates that the proposed method can ensure that fault tolerant cost of dynamically selected component is the locally optimal value, the function is to select the schema with the lowest cost in the execution process.

TABLE I
RESOURCE CONFIGURATION AND ATTRIBUTES

C	Actual meaning	CC	RA
$C_{1,1}$, $C_{2,1}$, $C_{3,1}$	Requirement analysis	99%	WS_1, WS_2, WS_3
$C_{1,2}$, $C_{4,1}$	Loading optimization	98%	WS_4, WS_5, WS_6
$C_{1,3}$, $C_{2,2}$, $C_{4,2}$	Route optimization	99%	WS_7, WS_8, WS_9
$C_{1,4}$, $C_{2,3}$, $C_{4,3}$	Network optimization	98%	WS_{10}, WS_{11}, WS_{12}
$C_{1,5}$, $C_{3,2}$, $C_{4,4}$	Payment completion	98%	WS_{13}, WS_{14}, WS_{15}
$C_{1,6}$, $C_{3,3}$	Transportation monitoring	97%	WS_{16}, WS_{17}, WS_{18}
$C_{1,7}$, $C_{3,4}$	Transportation querying	98%	WS_{19}, WS_{20}
$C_{3,5}$	Transportation route navigation	99%	WS_{21}
$C_{1,8}$,$C_{2,4}$,$C_{3,6}$	Late, and other service	97%	WS_{22}, WS_{23}, WS_{24}

TABLE II
ATTRIBUTE OF REPLACEMENT SERVICE

WS	ae	se	at	st	WS	ae	se	at	st
WS_1	2	3	2	1	WS_{13}	2	2	2	2
WS_2	3	2	2	2	WS_{14}	3	3	1	3
WS_3	2	2	2	3	WS_{15}	3	4	2	3
WS_4	4	2	2	1	WS_{16}	4	3	3	2
WS_5	5	4	1	2	WS_{17}	3	2	1	3
WS_6	2	2	2	2	WS_{18}	4	3	2	3
WS_7	2	3	2	3	WS_{19}	2	4	1	2
WS_8	3	2	3	3	WS_{20}	2	2	2	2
WS_9	3	3	1	1	WS_{21}	4	3	3	3
WS_{10}	4	2	2	2	WS_{22}	3	4	3	4
WS_{11}	2	3	2	3	WS_{23}	4	4	2	2
WS_{12}	2	3	3	3	WS_{24}	2	3	2	3

IV. EXAMPLE

In this paper, we use a simplified logistics cloud as an example. Four logistics clouds are operating at the same time, because each application has the different purpose, its execution processes are different too: $a_1 : C_{1,1} > C_{1,2} > (C_{1,3} \parallel C_{1,4}) > C_{1,5} > (C_{1,6} + C_{1,7}) > C_{1,8}$; $a_2 : C_{2,1} > (C_{2,2} \parallel C_{2,3}) > C_{2,4}$; $a_3 : C_{3,1} > C_{3,2} > (C_{3,3}+C_{3,4}+C_{3,5}) > C_{3,6}$; $a_4 : C_{4,1} > (C_{4,2} \parallel C_{4,3}) > C_{4,4}$. The attributes of component are shown in Table I. The weight of components in application is {0.1, 0.1, 0.2, 0.1, 0.1, 0.1, 0.2, 0.1}, {0.3, 0.3, 0.2, 0.2}, {0.2, 0.1, 0.2, 0.2, 0.1, 0.2}, {0.2, 0.3, 0.4, 0.1}. The system has 24 replacement services, their attributes are shown in Table II.

We can construct the fault tolerant model of replacement service, cloud application and cloud computing in the same way. We can verify the related properties of model by using the related tools of Petri nets, which includes the correctness of execution process and fault tolerant strategy, the selection of replacement service. Based on the state space of fault tolerant model, we can get that the state space is limited, and the applications can reliably operate when the component fails. We can randomly generate 15 fault tolerant strategies. i represents the service WS_i. First, we can compute the cost of $C_{1,1}$, $C_{2,4}$, $C_{3,6}$, $C_{3,5}$ under the different strategies, which is shown in Fig.3(a). We can get that the cost of $C_{2,4}$, $C_{3,6}$ under the same strategy is different even if the actual meaning and the replacement service of component are same. The cost

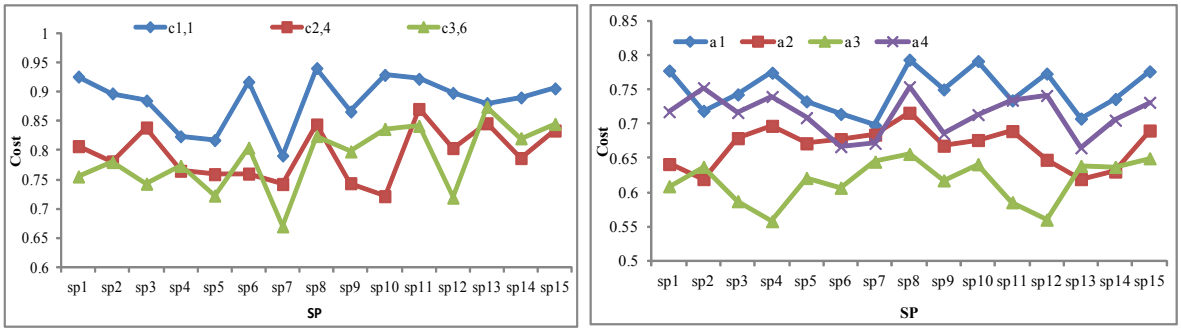

Fig. 3. Fault tolerant cost of application under different strategies

of all applications under the different strategy is shown in Fig.3(b). We can get that: (1) For the same application, the cost is different under the different strategies due to the cost of component is different. And all applications can get the lower cost under the strategy SP_8. (2) Not all applications can get the lowest cost under SP_8.

V. Related Works

Fault tolerance is an important means to improve the software reliability. Reference [6] proposes a large-scale fault injection system that can execute numerous model-based fault-injection simulations in a reasonable time using a cloud computing environment. A scalable hybrid Cloud infrastructure as well as resource provisioning policies to assure QoS targets of the users is presented in [7]. In order to increase the fault tolerance of cloud computing, a number of researcher and institutions begin to focus on cloud computing fault-tolerant framework [8], and to further explore the Byzantine fault -oriented cloud computing architecture [9]. However, several aspects differentiate our approach from the above approaches. First, the modeling and analysis process is easier to use because of the high abstraction level offered by using formal method, which help in strengthening the flexibility of composition process. Second, we propose the dynamic fault tolerant strategy, which can guarantee that cloud computing can select the optimal cloud service to realize the function of failed component, thus reducing the cost.

Many research efforts for cloud computing have adopted formal methods techniques to leverage its mathematically precise foundation for providing theoretically sound and correct formalisms. Bruneo, D. et al. propose a technique to model and evaluate the VMM aging process and to investigate the optimal rejuvenation policy that maximizes the VMM availability under variable workload conditions [10]. Ghosha et al. [11] develop a scalable stochastic analytic model for performance quantification of Infrastructure-as-a-Service (IaaS) Cloud. Reference [12] presents a framework called E-mc2 for modelling the energy consumption in cloud computing system. In contrast, we have proposed an approach to constructing the reliable service composition, which provides means to observe behaviors of basic component, and to describe their interrelationship [13]. Most of the aforementioned formalisms cover basic and structured activities of cloud application, but they are unable to ensure that the constructed model can meet the users' requirements, such as cost and reliability.

VI. Conclusion

In this paper, Petri nets are used to describe different components of cloud computing. The reliability and cost are took into account in the modeling process. Then, we propose a method to dynamically make fault tolerant strategy, which can get the fault tolerant strategy with the lowest cost based on the current state and failed component. Third, we present the operational semantics and related theories of Petri nets to help prove the effectiveness of proposed method,. Finally, we also conduct experiments to evaluate the proposed method.

Acknowledgment

The work is partially supported by the NSF of China under grants No. 61173048 and 61300041. Research Fund for the Doctoral Program of Higher Education of China under Grants No. 20130074110015.

References

[1] S. Marstona, Z. Lia, S. Bandyopadhyaya, et al. Cloud computing-the business perspective. Decision Support Systems. 2011, 51(1): 176-189.
[2] B. Yang, F. Tan, Y. S. Dai. Performance evaluation of cloud service considering fault recovery. The Journal of Supercomputing. 2013, 65(1):426-444.
[3] Wang, X. Y, Du, Z. H, Chen, Y. N. An adaptive model-free resource and power management approach for multi-tier cloud environments. Journal of Systems and Software. 2012, 85(5): 1135-1146.
[4] M. TADAO. Petri nets: properties, analysis and application. Proceedings of the IEEE. 1989, 77(4):540-581.
[5] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, R. Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience. 2011, 41(1):23-50.
[6] Y. Nakata, Y. Ito, Y. Takeuchi, et al. Model-based fault injection for large-scale failure effect analysis with 600-node cloud computers. http://cs28.cs.kobe-u.ac.jp/assets/files/pdf/1303_nakata_RIIF.pdf.
[7] B.Javadi, J. H. Abawajy, R. Buyya. Failure-aware resource provisioning for hybrid Cloud infrastructure. Journal of Parallel and Distributed Computing. 2012, 72(10): 1318-1331.
[8] G. Belalem, S. Limam. Fault tolerant architecture to cloud computing using adaptive checkpoint. International Journal of Cloud Applications and Computing. 2011, 1(4): 60-69.
[9] Y. Zhang, Z. Zheng, M. R. Lyu. BFTCloud: A byzantine fault tolerance framework for voluntary-resource cloud computing. Processing of the 2011 IEEE International Conference on Cloud Computing. IEEE Computer Society, Washington, DC, USA, 2011: 444-451.
[10] D. Bruneo, S. Distefano, F. Longo, A. Puliafito, M. Scarpa. Workload-based software rejuvenation in cloud systems. IEEE Transactions on Computers. 2013, 62(6):1072-1085.
[11] R. Ghosha, F. Longob, V. K. Naikc, K. S. Trivedi. Modeling and performance analysis of large scale IaaS Clouds. Future Generation Computer Systems. 2013, 29(5):1216-1234.
[12] G. C. Gabriel, N. Alberto, L. Pablo, et al. E-mc2: A formal framework for energy modelling in cloud computing. Simulation Modelling Practice and Theory. 2013, 39(2013): 56-75.
[13] G. Fan, H.Yu, L.Chen, D.Liu. Petri net based techniques for constructing reliable service composition. Journal of Systems and Software. 2013, 86(4): 1089-1106.