2011 International Conference on Computer and Software Modeling
IPCSIT vol.14 (2011) © (2011) IACSIT Press, Singapore

Simulation on Availability Analysis of Reconfigurable Software Systems

Chaoqun Zhang, Minyan Lu and Lingzhong Meng
School of Reliability and System Engineering, Beihang University, Beijing, China
superme0308@gmail.com, lmy@buaa.edu.cn, menglz@dse.buaa.edu.cn

Abstract. Reconfigurable software systems are usually responsible for realizing many kinds of client requests. As the complexity of such systems architecture, using mathematical analysis is difficult to directly analyze its availability. Therefore, this paper describes a simulation method to analysis the availability of reconfigurable software systems. Firstly, based on structural features of the reconfigurable software systems, build the system model, then simulate running states of software systems, and build the dynamic process of systems. Finally, monitor state variables in the simulation to analyze system availability.

Keywords: software availability, reconfigurable software systems, system modeling

1. Introduction

With the diversification of user requirements, and in order to improve software usability, configurable software systems have become a trend. In order to improve software availability and configurability [1], configurable software systems is widely applied, users can configure the software system availability and system availability metrics more and more attention.

Now, more reconfigurable software systems used in network software. Such software systems ability to provide services through the network has great importance. Therefore, the protection of the availability of such software systems and robustness is very necessary. For operators, the system is running, it is necessary to consider the availability as an important factor [2].

However, the configurable architecture of software systems failure in the system can be reconfigured so that the availability analysis of software systems becomes complex, difficult to direct its usability studies [3, 4]. To this end, we propose a configurable software system for modeling and simulation, monitoring of state variables in the simulation process, in order to analyze the availability of configurable software systems.

2. Reconfigurable Software Systems Model

Reconfigurable software systems is the software system failure occurs when the system is re-configured for rapid re-configuration. Through the re-configuration can not only improve system availability, making the face a variety of failure, local conditions enable the appropriate configuration [5], and in some special cases, can make the system function has more to do a completely different change tasks.

Most of these systems is the use of modular design, composed by the elements. Because with the large-scale parallel computing and distributed programming technologies, the phenomenon of software modules works together more and more. Individual components can only complete a simple function, but in practical applications, often require multiple components together to complete a service. If the function of a component failure occurs, the corresponding service can’t be completed. So the system is mostly composed of multiple components.

Corresponding author. Tel.: +86-13810804393
E-mail address: superme0308@gmail.com

The system can meet the needs of users at various levels, with a rapid response capability: a software system can be configured a variety of user needs, in order to adapt to different user needs, we must improve the system software from the configurability [6]. In addition, the system must be based on changes in the environment, adopt appropriate strategies to achieve the appropriate response to the external environment.

Software system can be configured to be applied more network software. Assuming such network software system is composed by a series of nodes, nodes and paths between nodes form the basis of the entire network software system for the performance of the system is very important. Building the system model, nodes and paths between nodes must be considered.

Therefore, when considering the configuration of the system, must be routes between nodes and nodes as the basic element. Among them, Ni represents the system nodes, where i implies the node number. Rij represents the routes between Ni and Nj. And, Pi represents the failure rate of nodes or routes. Therefore, the software systems configuration is a collection of interconnected nodes and routes, which is represented as W, where W can be described as:

$$W = \langle [N_i\text{：} N_i \in N]\text{，} R\rangle \tag{1}$$

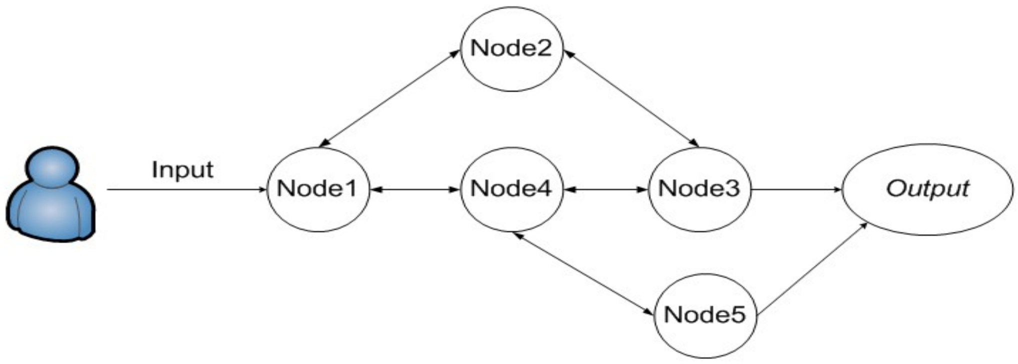

Fig.1 Example of Reconfigurable Software Systems Model

Shown in Figure 1, the software system configuration, user node1 to accept user's input request, and use node 3 to offer the services. In the example users connect to the service via node 1, connection 1-4, node 4 and connection 3-4. Thus, the initial configuration of the system is given by:

$$W1 = \left\langle \begin{matrix} N = [N1\text{，} N3\text{，} N4]\text{，} \\ R = \{\{(N1\text{，} N4)\text{，} (N4\text{，} N3)\}\text{，} \{(N2\text{，} N4)\}\text{，} \{(N1\text{，} N2)\}\} \end{matrix} \right\rangle \tag{2}$$

3. Software systems reconfiguration process

For the software systems model, it is necessary to describe the failure of logic. The failure of logic used to determine whether the failure of a node failure affects the whole mission, as a key node into the node and non-critical node; when the critical node failure, can lead to the failure of the entire mission [6].The non-critical node is the node failure occurs, but also by re-configuring other nodes to complete the call to the appropriate function module. In the above systems, N1 failure will cause the entire system can’t complete its function, therefore, the software system, N1 as the key node. N3 provides server functionality but can have other nodes to complete, so N3 is non-critical node.

The software systems can be configured when the software is some non-critical nodes failure occurs, the system reconfiguration, change the original configuration, making the occurrence of failure to timely update the system based configuration and restore its function. For example, a node occurs when the system fails, the system provides functionality to change the node from other nodes to provide [7]. This time, the system configuration becomes W ':

$$W = \langle [N_i\text{：} N_i \in N]\text{，} R\rangle \rightarrow W' = \langle [N_i\text{：} N_i \neq \text{failed}]\text{，} R'\rangle \tag{3}$$

Where R’ is the set of routes obtained from R by eliminating all those passing through failure nodes.

When the route failures, the software systems configuration becomes W’’:

$$W = \langle [N_i\text{：} N_i \in N]\text{，} R\rangle \rightarrow W'' = \langle [N_i\text{：} N_i \in N]\text{，} R''\rangle \tag{4}$$

Where R’’ is the set of routes obtained from R by eliminating all failure routes.

As a result of failure of node 3, the system degrades to the following, obtained by applying (3) to the above:

$$W2 = \left\langle \begin{matrix} N = [N1,\ N2,\ N4], \\ R = \{\{(N1,\ N4)\},\ \{(N2,\ N4)\},\ \{(N1,\ N2)\}\} \end{matrix} \right\rangle \quad (5)$$

By similar reasoning, it is possible to determine the configuration when connection 1-4 fails and the system is rerouted to 1-2 and 2-4, given by equation (6). It is also possible to reconfigure the system to accommodate both the failure of node 3 and of connection 1-2. This is given by equation (7). It should be noted that the failures do not affect the routes necessary for reconfiguration, so the service is kept available.

$$W3 = \left\langle \begin{matrix} N = [N1,\ N2,\ N3,\ N4,\ N5], \\ R = \{\{(N1,\ N2),\ (N2,\ N3)\},\ \{(N2,\ N4)\},\ \{(N1,\ N2),\ (N4,\ N5)\}\} \end{matrix} \right\rangle \quad (6)$$

$$W4 = \left\langle \begin{matrix} N = [N1,\ N2,\ N4,\ N5], \\ R = \{\{(N1,\ N2),\ (N2,\ N4)\},\ \{(N2,\ N4)\},\ \{(N1,\ N2),\ (N4,\ N5)\}\} \end{matrix} \right\rangle \quad (7)$$

$$W5 = \left\langle \begin{matrix} N = [N1,\ N2,\ N3], \\ R = \{\{(N1,\ N2),\ (N2,\ N3)\},\ \{(N1,\ N2),\ (N4,\ N5)\}\} \end{matrix} \right\rangle \quad (8)$$

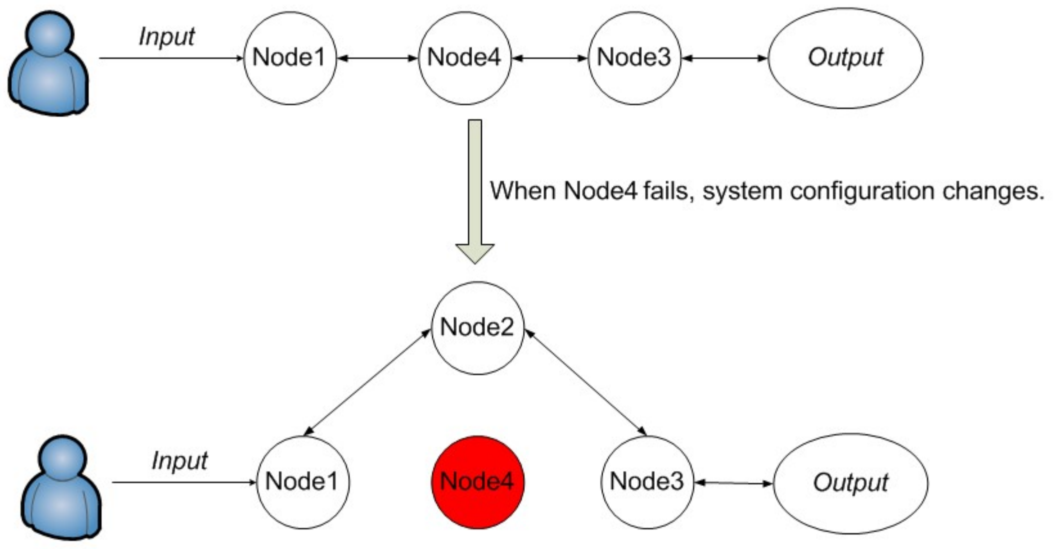

Fig.2 Example of reconfiguration of software systems

4. Simulation of Software Systems

In the reconfigurable software system as shown in Figure 1, the failure probability of nodes and routes is random. Assuming the failure rates of nodes and paths are equal (exponentially distributed). And the nodes and the routes are known as system components.

1) Assuming the software system and path of the node number n, the simulation run time T=0, it can produce n-sample of the basic components of the failure time, small to large order is:
$t1 < t2 < t3 < \cdots < tn$
(8)
2) Put the failure parts into failure states in the small to large order of the failure times, and then determine whether the key component parts, if key components, the system failure; if not the key components, then under state transition can be configured, the system is updated to the new configuration, and continue to run the simulation [7].
3) If the simulation process, the failure of essential components have led to system failure. Assume that the system repair time which is represented as t_fix obey the positive normal distribution. Reconfiguration is again performed after the system is repaired, to bring back the original configuration. And run the simulation until the simulation time to run to the specified value TIME.

Through the simulation, the systems availability can be gained, which is presented as A:

$$A = \frac{T\_s}{TIME} \quad (9)$$

Where T_s is the systems successfully running time, and TIME means the full simulation time. The assumption that the various components of the system failure rate in exponential distribution, so the failure of systems has a certain randomness.

Reconfigurable software system shown in Figure 1, running simulation program, based on changes in the failure rate can be used to compare the level of availability. And the simulation parameters of reconfigurable software systems are shown in table 1, with the systems running time is 100 minutes:

Tab.1 Availability Parameters of Reconfigurable Software Systems

Unit	Number of failures	Time of failure(min)	Successfully Running time(min)	Availability(%)
Node1	7	6.77	93.23	0.9323
Node2	4	3.56	96.44	0.9644
Node3	13	15.97	84.03	0.8403
Node4	17	20.20	79.80	0.7980
Node5	7	7.54	92.46	0.9246

And also, through the simulation, the availability of the example systems with the change of the running time can be get, which is shown in figure 3.

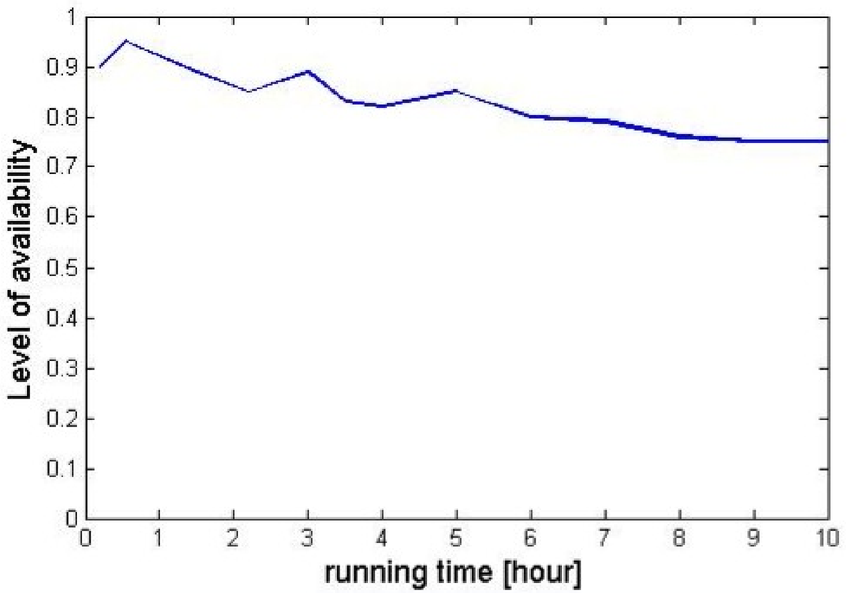

Fig.3 Availability of the example reconfigurable software systems

5. Conclusions

Reconfigurable software systems, because of its complexity and dynamics, it is difficult to measure using the method of mathematical analysis of the availability of such systems. And because the system failure is going to avoid as much as possible, the simulation method can reduce the resources and manpower expenses. So early in the design of software systems, developers can use simulation methods to assess the system availability.

In engineering practice, the simulation analysis of software systems can help to find a good re-configuration as possible to reduce or avoid software failures may occur. So this method can be used to optimize the system configuration, improve system availability, enhance system maintainability.

6. References

[1] A. Avizienis, J. Laprie and B. Randell, *Fundamental Concepts of Dependability*, Research Report.N01145, LAAS-CNRS, 2000

[2] D. M. Nicol, W. H. Sanders and K. S. Trivedi, Model-based evaluation from dependability to security. *Dependable and Secure Computing*, IEEE Transac- tions on Volume 1, Issue 1, Jan 2004: pp.48-65

[3] J. Erickson, *Hacking; The Art Of Exploitation Hacking.* San Francisco, CA: No Starch Press. 2003:pp.101-103.

[4] M. Erbschloe, Trojans, Worms and Spyware; A Computer Security Professionals Guide to Malicious Code. Oxford: Butterworth-Heinemann. 2004:232-235

[5] D. Caban. The impact of software on overall system dependability. *Computer Engineering* (ed.W. Zamojski), WKŁ, Warsaw. 2005:pp.12-17

[6] T. Li and Y. W Dong. A Component Model and Component Repository for Embedded Software. *Computer Science*. 2006, 33(11)

[7] W. Zamojski. Functional reliability model of a man-computer system. *CollectiveWorks: Computer Engineering,*WKiŁ, Warsaw 2005