Automatic Diagnosis for Composite Web Service in Cloud Computing

Zhichun Jia, Rong Chen, Junjie Xu
School of Information Science and Technology
Dalian Maritime University
Dalian, Liaoning 116026, China
zhichun.jia@dlmu.edu.cn, rchen@dl.cn, connyadmin@163.com

***Abstract*—Cloud computing is becoming a popular and important platform for web service applications. With the advent of more and more service resources in cloud, it is a crucial challenge to build an effective diagnostic mechanism. To improve the diagnosis capability for composite web service, we propose an automatic diagnostic framework and a hybrid diagnosis method for diagnosing the faulty activities. Our diagnostic framework can improve the scalability of diagnostic system and add the diagnostic capability of diagnosis service by decoupling the diagnostic service components. Our diagnosis method identifies the differences between successful and failed executions for finding the incorrect activities of the composite service and provides faulty causes by considering the structured activities in the process of diagnosis. Experimental results show that our method is effective to the composite web service in the fault diagnosis.**

Keywords- fault diagnosis; composite web service; cloud computing; activity dependence graph

I. INTRODUCTION

As an important and popular platform of service integration, cloud computing provide scalable utilization of massive service resources. With the exponential growth of service resources, one of the difficult challenges in cloud computing is how to locate and remove unreliable services from web service composition.

For achieving reliable service composition, the diagnosis and handling methods for faults in the web service compositions are desirable. At present, there are two major methods for diagnosing service faults: the model-based (MBD) method [1-10] and process history based method [11-13]. MBD method models the web service composition by the behavior and function specification of service. According to the formal models and the runtime observations during the service execution, MBD method can locate the faulty behaviors and explain the faulty root causes. Service execution history based method uses the historical data as its priori knowledge to diagnose the faulty service. However, these two methods aren't adequate to meet the growing requirements of the service composition processes for a diagnostic system in the large-scale and complex computing environment.

To address this critical challenge, we propose a hybrid method for diagnosing faulty services. In our method, we combine historical execution data with the dependence graph model of service composition. For identifying the incorrect activities, we compute the similarity between activity dependence and faulty behaviors by the differences between successful and failed executions of composite service. The main contributions of this paper include the followings:

- We present a diagnostic architecture for automatically diagnosing faults of web service compositions in the cloud computing environment.
- We propose a novel diagnosis method for locating the multi-faults in the web service composition, where we compute the activity dependence similarity according to the dependence graph model and historical execution data. Finally, we can rank the suspicious activities by their suspiciousness scores.
- We conduct the simulation experiments based on 2507 real web services. Experimental results have verified the effectiveness of our method.

The rest of paper is organized as follows. Section 2 introduces the framework of diagnostic system. Section 3 presents the activity dependence graph. Section 4 proposes the diagnosis method. Section 5 describes our experiments, and shows our experimental results. Finally, section 6 gives the conclusion.

II. FRAMEWORK OF DIAGNOSTIC SYSTEM

For facilitating diagnosis, we assume that a description of composite web service process, such as BPEL (Business Process Execution Language) file, is known and historical data of service execution are kept in the composite service execution log such that:

- executed activities and the messages of input and output are available in the composite service execution log and recorded in executed order;
- the composite service execution log contains execution traces and the execution results (success or failure) of a composite service process.

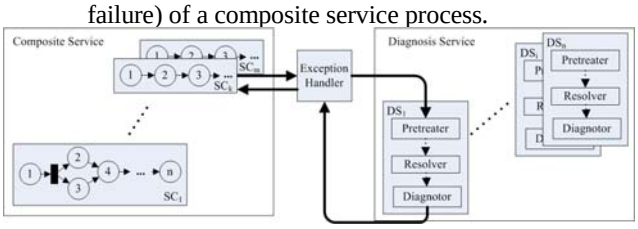

Figure 1. Framework of diagnostic system.

Our framework of diagnostic system shown in Figure 1 mainly includes the following parts:

- A faulty web service composition SC_i throws exception during the execution.
- The exception handler catches the exception of SC_i. Then it selects an appropriate diagnosis service DS_j to trigger it.
- The pretreater of DS_j receives the diagnostic request from the exception handler and respectively generates the activity dependence graph and test cases according to the description file and execution log of SC_i.
- The resolver of DS_j analyzes the similarity between activity dependences and faulty behaviors and provides a similarity coefficient for each activity dependence relation.
- The diagnotor of DS_j uses the similarity coefficient provided by the resolver to compute and assign a suspiciousness value to each activity. By ranking these suspiciousness values, the diagnotor sends a diagnostic result to the exception handler for recovering the composite web service process SC_i.

III. Activity Dependence Graph (ADG)

Before describing the activity dependence graph, we firstly give the definition of activity dependence.

Definition 1.1 Activity Dependence: An activity t_j is activity dependent on another activity t_i iff the next execution activity of t_i is t_j in at least one execution trace.

An activity dependence graph is created according to the dependence relations between the activities. The activity dependence graph is described as follow.

Definition 1.2 Activity Dependence Graph: An activity dependence graph is a 3-tuple (*Nb*, *Ns*, *E*), where:

- *Nb* is a set of nodes that represent the basic activities in the web service composition;
- *Ns* is a set of nodes that represent the structured activities in the web service composition;
- *E* is a set of directed edges, edge (n_i, n_j) represents that node n_j is dependent on node n_i.

An activity dependence graph contains two types of nodes. Basic activity nodes, depicted as round, represent the basic activities of the web service composition. Structured activity nodes, represented as rectangles, correspond to the structured activities. Activity dependence edges, shown as solid lines with arrow, represent the activity dependences. To illustrate, consider the example shown in Figure 2.

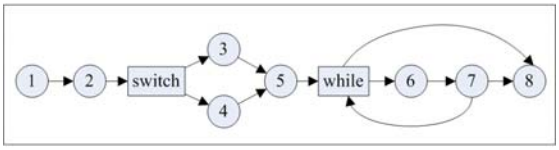

Figure 2. An example of the activity dependence graph.

In Figure 2, nodes 1-8 represent the basic activities and nodes *switch* and *while* represent the structured activities. Edges (*switch*, 3) and (*switch*, 4) represent node 3 and node 4 are activity dependent on node *switch*.

IV. Diagnosis Model

In Section 2, we have briefly introduced our diagnosis model. Our diagnosis model mainly includes three parts: pretreater, resolver and diagnotor.

A. Pretreater

The goal of pretreater is to respectively generate the activity dependence graph and test cases according to the description file and execution log of faulty web service composition. In Section 3, we have described the generation of the activity dependence graph. Here, we primarily discuss how to generate the test cases.

If an execution trace contains the nodes, which are activity dependent on a structured activity, and doesn't contain this structured activity, we insert a dummy structured node before these nodes. For example, if an execution trace for the example in Figure 2 is (1, 2, 3 …), we insert a dummy node *switch* before node 3.

Moreover, in BPEL there is a special dependence relation, which occurs in a *flow* activity. For this concurrent dependence relation, several activities are dependent on an activity *flow* in the same execution trace. Hence, the sequence of these concurrent activities may be different in the execution traces. The discussion of the concurrent activities is out of the scope of this paper.

After inserting the structured nodes, we convert each execution log into a test case. Let $EC = \{tc_1, tc_2, \cdots, tc_n\}$ be a test suite for a faulty web service composition, where tc_k ($1 \le k \le n$) is a test case. *EC* can be divided into two disjoint subsets EC_0 and EC_1 according to the execution results (success is 0 or failure is 1) of test cases. Given an activity dependence $ad = (a_1, a_2)$, we say that a_2 is activity dependent on a_1. Let $EC(ad)$ be a set of test cases, where each test case covers the activity dependence *ad*.

B. Resolver

The similarity between activity dependence and faulty executions is computed by a similarity coefficient. The similarity indicates the correlation between the occurrences of an activity dependence and observed incorrect behaviours [14]. At present, there are many similarity coefficients proposed, such as Tarantula [15], SBI [16] and Ochiai [17]. We use the similarity coefficient in Ochiai to compute the activity dependence similarity.

$$sim(ad) = \frac{|EC_1(ad)|}{\sqrt{(|EC_1(ad)| + |EC_0(ad)|) \times |EC_1|}}. \quad (1)$$

C. Modeler

For the basic activities, we use the average value of all dependence similarities of an activity as the suspiciousness value of this activity. If an activity isn't contained in any dependence, the suspiciousness value of this activity is 0.

$$sv(a) = \frac{\sum sim(\theta)}{\sum |\theta|}, \theta \in ad(a,\bullet) \bigcup ad(\bullet,a). \quad (2)$$

For the structured activities, we use (3) to compute their suspiciousness values.

$$sv(a) = \sqrt{\frac{\sum_{i=1, j=i+1}^{n} (sim(ad(a,a_i)) - sim(ad(a,a_j)))^2}{n-1}}. \quad (3)$$

D. Algorithm

For locating the fault in the web service composition, we compute the suspiciousness score of each activity and give a diagnostic result by using Algorithm 1.

Algorithm 1: **Activity Suspiciousness Algorithm**

Input: specification *Sp* and execution log *El* of faulty web service composition

Output: diagnostic result *DR*

1: to read *Sp* and *El*, generate activity dependence graph *adg* and test cases *TS*

2: for each activity dependence $(a_i,a_j) \in adg.E$, compute $sim(a_i,a_j)$ by (1)

3: for each activity $a_i \in adg.Nb \cup adg.Ns$, compute $sv(a_i)$ by (2) and (3)

4: to rank $sv(a_i)$ in descending order in *DR*, where $a_i \in adg.Nb \cup adg.Ns$

5: return *DR*

V. Experiments

In our simulation experimental environment, there are three parts. Part one is to build a workflow according to the given numbers of activities and structured nodes. Part two is to generate the execution traces of this workflow according to real service property and given number. In part one, we assign a set of service properties to every service. These properties are from the QWS database [18], which collected 2507 web services and performed various measurements, such as response time, availability and reliability. Part three is our diagnosis algorithm. Figure 3 depicts our framework of simulation experiment.

To evaluate the effectiveness of our method, we generate 10 workflows for 10 test groups. Each test group has the same number of activities. For each workflow, we generate 200 test cases and 50 faulty traces to be diagnosed. Then we compute the rate of the number of correct diagnosis to the number of faulty traces for each workflow, and use this rate as the accuracy of this workflow. Here, we respectively consider the top 1-5 suspicious activities as a correct diagnosis. The accuracy for each test group is the average of the accuracies of 10 workflows in this test group.

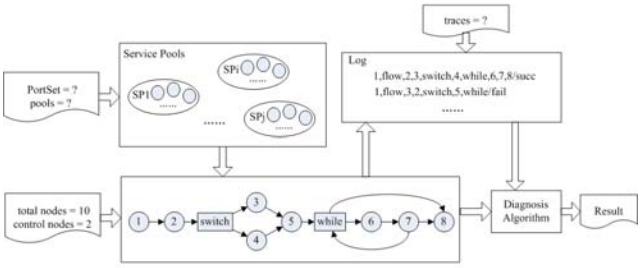

Figure 3. Framework of simulation experimental environment.

TABLE I. Accuracy of top-1 suspicious activity

Node	1	2	3	4	5	6	7	8	9	10	Average
10	0.66	0.71	0.86	0.73	0.74	0.97	0.73	0.84	0.66	0.71	0.76
20	0.78	0.97	0.74	0.86	1.00	0.68	0.82	0.60	0.95	0.93	0.83
30	0.95	0.99	0.96	0.95	0.72	0.86	0.70	0.72	1.00	0.74	0.86
40	1.00	0.74	0.79	0.57	0.73	0.78	0.79	0.77	0.81	0.89	0.79
50	0.74	0.82	0.64	1.00	0.91	0.79	0.82	0.56	0.80	0.79	0.79
60	0.67	0.83	0.69	0.66	0.89	0.77	0.79	0.55	0.95	0.73	0.75
70	1.00	0.79	0.97	0.67	0.59	1.00	1.00	0.71	0.71	0.71	0.82
80	0.82	0.81	0.96	0.81	0.60	0.80	0.94	0.88	0.80	0.75	0.82
90	0.90	0.83	0.76	0.76	0.75	0.74	0.79	0.78	0.79	0.82	0.79
100	0.70	0.68	0.95	0.68	0.88	0.80	0.71	0.70	0.77	0.78	0.77

Table 1 shows that the accuracy of top-1 suspicious activity as a diagnosis for each workflow. The first column denotes the number of activities in a workflow. The last column denotes the average of accuracies for each test group. We can see that the accuracy of our diagnosis method is above 76%.

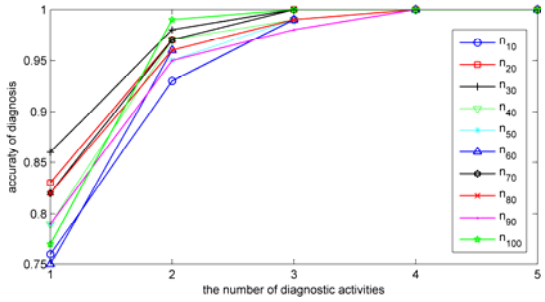

Figure 4. Accuracies for top-1 to top-5 suspicious activities as the diagnostic results.

Figure 4 shows the accuracy of our method is almost 100% when we consider top-4 suspicious activities as diagnosis. The experiments show our method is effective for diagnosing the faulty web service compositions. Here, n_x denotes the test group that all workflows both have x activities.

VI. Related Works

Our idea is inspired by spectra-based fault localization (SFL), which is a well known technique in program fault diagnosis. Many efforts have been devoted to develop SFL techniques, such as Turantula [15], Ochiai [17] and SBI [16].

These methods use different techniques to assign a value for each program statement's likelihood of being faulty. LOUPE method not only used similarities to assess the statement suspiciousness but also built different models for different types of spectra. However, several activities can execute concurrently in web service composition, while program statements execute in order. Hence, the dependence relation of web service composition is more complex than that of program.

In order to enhance fault management in complex web services, Ardissono [6] proposed a diagnostic framework for adding diagnostic capabilities to web services, and a hierarchical MBD method. However, this method isn't suitable for diagnosing the service faults under the large-scale and complex computing environment when most of the web services claim too coarsely. Li [4] proposed a diagnosis method for web services in a choreographed scenario, where local fault models are converted into a CPN (colored Petri nets) model. The diagnostic effectiveness of this method depends on the integrity of fault model. However, many faults of web service are generally unpredictable.

In this paper, we consider historical data into the service model for overcoming the shortcomings of above methods.

VII. CONCLUSION

In this paper we present a diagnostic framework and a hybrid method to automatically diagnose the faults during the execution of web service composition. Scalability is a feature of our method which allows diagnosis service to integrate functional components (preprocessor, analyzer and modeler) as a further means to deal with different complex web service compositions. The activity dependence graph-based diagnosis is another important feature of our method which uses the differences between successful and failed executions to identify the incorrect activities. Experimental results show that our method is effective.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (61175056), the Dalian Maritime University Backbone Youth Foundation (NO.2011QN033, No.2009JC29), and IT Industry Development of Jilin Province.

REFERENCES

[1] Y. Yan, P. Dague, Y. Pencole , and M.-O. Cordier, "A Model-based Approach for Diagnosing Faults in Web Service Processes," The International Journal of Web Services Research (JWSR), vol. 6, pp. 87-110, 2009.

[2] L. Ardissono, S. Bocconi, L. Console, R. Furnari, A. Goy, G. Petrone, et al., "Enhancing Web Service Composition by Means of Diagnosis," Business Process Management Workshops, vol. 17, pp. 468-479, 2008.

[3] W. Mayer, G. Friedrich , and M. Stumptner, "Diagnosis of Service Failures by Trace Analysis with Partial Knowledge," Service-Oriented Computing, vol. 6470, pp. 334-349, 2010.

[4] Y. Li, L. Ye, P. Dague , and T. Melliti, "A Decentralized Model-Based Diagnosis for BPEL Services," Proceedings of the 2009 21st IEEE International Conference on Tools with Artificial Intelligence (ICTAI '09), pp. 609-616, 2009.

[5] M. Alodib and B. Bordbar, "A model-based approach to Fault diagnosis in Service oriented Architectures," ECOWS'09 - 7th IEEE European Conference on Web Services, pp. 129-138, 2009.

[6] L. Ardissono, L. Console, A. Goy, G. Petrone, C. Picardi, M. Segnan, et al., "Cooperative Model-Based Diagnosis of Web Services," Proceedings of 16th International Workshop on Principles of Diagnosis 2005.

[7] X. Han, Z. Shi, W. Niu, K. Chen , and X. Yang, "Similarity-Based Bayesian Learning from Semi-structured Log Files for Fault Diagnosis of Web Services," presented at the Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on, 2010.

[8] S. C. Hui, A. C. M. Fong , and G. Jha, "A web-based intelligent fault diagnosis system for customer service support," Engineering Applications of Artificial Intelligence, vol. 14, pp. 537-548, 2001.

[9] F. Moo-Mena, J. Garcilazo-Ortiz, L. Basto-Diaz, F. Curi-Quintal , and F. Alonzo-Canul, Defining a Self-Healing QoS-based Infrastructure for Web Services Applications. Los Alamitos: Ieee Computer Soc, 2008.

[10] L. Wang, X. Bai, L. Zhou , and Y. Chen, "A Hierarchical Reliability Model of Service-Based Software System," presented at the Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International, 2009.

[11] H. N. Lakshmi and H. Mohanty, "Automata for Web Services Fault Monitoring and Diagnosis," IJCCT, vol. 3, pp. 13-18, 2011.

[12] Y. Dai, L. Yang, B. Zhang , and Z. Zhu, "Exception diagnosis for composite service based on error propagation degree," 2011 IEEE International Conference on Services Computing, SCC 2011, pp. 160-167, 2011.

[13] Z. Zhu and W. Dou, "QoS-Based Probabilistic Fault-Diagnosis Method for Exception Handling," New Horizons in Web-Based Learning: Icwl 2010 Workshops, vol. 6537, pp. 227-236, 2011.

[14] K. Yu, M. Lin, Q. Gao, H. Zhang , and X. Zhang, "Locating faults using multiple spectra-specific models," in Proceedings of the 2011 ACM Symposium on Applied Computing, TaiChung, Taiwan, 2011, pp. 1404-1410.

[15] J. A. Jones and M. J. Harrold, "Empirical evaluation of the tarantula automatic fault-localization technique," in Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, Long Beach, CA, USA, 2005, pp. 273-282.

[16] Y. Yu, J. A. Jones , and M. J. Harrold, "An empirical study of the effects of test-suite reduction on fault localization," presented at the Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on, 2008.

[17] R. Abreu, P. Zoeteweij, R. Golsteijn , and A. J. C. van Gemund, "A practical evaluation of spectrum-based fault localization," Journal of Systems and Software, vol. 82, pp. 1780-1792, 2009.

[18] E. Al-Masri and Q. H. Mahmoud, "Investigating web services on the world wide web," in Proceedings of the 17th international conference on World Wide Web, Beijing, China, 2008, pp. 795-804.