Service Selection Using Service Clusters

ZhengZhe Xiang, Shuiguang Deng, Honghao Gao
College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China
{xiangzhengzhe,dengsg,gaohonghao}@zju.edu.cn

***Abstract*—With the rapid development of cloud computing technologies, there are increasing number of Web services deployed by service providers on cloud platforms. Invoking Web services is in the form of given orders, by which the end-user can process business and information in anytime and anyway. However, existing Web service compositions mainly focus on single Web services selection and neglect to consider the fact that complex services created and aggregated by service compositions might be low global QoS (Quality of service). In this paper, the importance and necessity to construct service clusters in business process are discussed. Then, how to build up a model to solve the optimal QoS problem of Web service composition using service clusters is proposed. Third, the dynamic programming algorithm is introduced to this model. Finally, a series of experiments are carried out to evaluate our method. The results show that service cluster has advantage to obtain composite service with best QoS for the end-user**.

Keywords—service composition, service cluster, service selection

I. INTRODUCTION

A. Background

Web service is known as one of the most popular techniques to build distributed systems [1] for its platform independence and loose coupling. However, the single Web service can't satisfy complex requirements. The end-users should interact with cloud frequently and invoke the target service continuously. To this purpose, the Web service composition technology has been proposed and widely used by the academy and industry. It provides a promising solution to integrate single-functional services seamlessly to create a new large-granularity and value-added service [2]. Component services are selected according to the non-functional and functional characteristics. Then, the execution orders are arranged in the light of the workflow specified by the end-user.

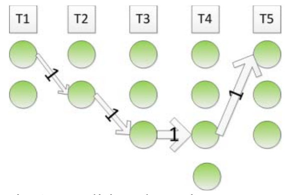

Fig.1. Traditional service composition model

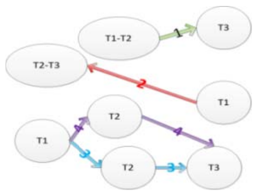

Fig.2. Existence form of services in real word

QoS describes the non-functional characteristics of services, such as execution time, cost and availability. Traditional service composition problem is QoS-aware optimal problem. Given a workflow of certain execution orders and sets of candidate services with same atomic functionalities, the service composition tries to assemble individual services under constraints of QoS requirement [3]. But, the existence form of services on Internet has various presentations, which have produced inconsistent forms compared with template descriptions since it contains complex functionalities. As Fig.1 shown, typical service form is considered by previous works, but they do not consider the form shown in Fig.2 which will impact the global QoS a lot.

B. Motivation

Web services with single or atomic functionality is used as basic components during Web service selection. But, it may be a tunnel vision in the development of service computing technology and big data research. As a service provider, the service business community's goal is to get benefits by attracting more end-users to use his platform. Analyzing the requirements of clients to satisfy them is the first important task. Various methods have been proposed to achieve this goal. However, the most efficient one is process mining and merging. Table I gives an example of services provided by Amazon, where ITT means the input parameters transmission time, ET means service execution time, OTT means output parameters transmission time and RT means the total running time.

TABLE. I THE TIME COST OF SERVICES PROVIDED BY AMAZON

WS	Provider	RT(ms)	ITT(ms)	ET(ms)	OTT(ms)
S1	Amazon	328	173	43	112
S2	Amazon	310	145	41	124

After a running period, Amazon finds that, Web service S2 comes after S1 at 90%. Thus, they create a complex service S4 as a service cluster including the functionality that S1 and S2 offer. S4 is not an integral composite service but a service fragment. If S4 is involved, the execution orders of workflow should be changed. Due to the service providers come from the same company, the transmission between platform and end-user is omitted, which reduces the total run time during service invoking. The time cost of workflow S1->S2 is not S1.RT+S2.RT=638ms, but S4.RT= S1.ITT+S1.ET+S2.ET+S2.OTT=381ms. The improvement is remarkable. Under this improvement, the end-user will prefer to select service clusters provided by Amazon than others.

A detailed case is shown in Fig.3. The green circles stand for atomic services with single functionality, and the yellow circles with rectangle contain service clusters. For the latter, the end-user can get better service composition (QoS.exetime=18.6ms) than traditional approach (QoS.exetime = 20.0ms).

II. Related Work

As mentioned above, service selection is the main problem met in service composition. In the field of service selection, the previous work is QoS-aware. Atomic service with single functionality is used as basic component, and QoS is considered as the non-functional characteristic. Different methods are proposed to solve it from different perspective [4-8]. Some of them treat it as a local optimal problem and deal it with greedy strategy [9-11]. Some of them study different categories of QoS and use multi-objective decision method [12-15]. However, it lacks to consider the service selection from cloud platform provider way and service cluster for building complex services.

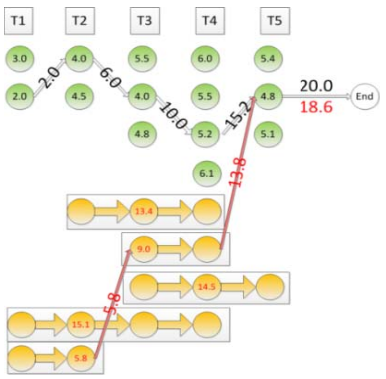

Fig. 3 A Detailed Case

III. Problem Definition

In this section, we formally definite the service composition problem using service cluster. The basic concepts are first to be introduced.

Definition 1(Web Service): A Web service is a 2-tuple (*func, QoS*), where:

(1) *func* is the functionality of Web service

(2) *QoS* is the quality of service (QoS), including execution time, cost, reputation, reliability, etc.

In Definition 1, the Web service is represented by its functionality and QoS. There would be lots of services with the same functionality, but they can be distinguished by QoS characteristics. It is priority to select the one with better QoS.

Definition 2(Service Cluster): A service cluster is a 2-tuple (*funcs,QoS*), where:

(1) *funcs* is a list of sequential tasks that the service cluster will execute orderly.

(2) *QoS* is the quality of service cluster.

In Definition 2, a service cluster is represented by its functionality set and QoS. Once a service cluster is invoked by an end-user, it will be terminated until all tasks have been done. It is an integration without dividable.

Definition 3(Service Plan): A service plan is a 2-tuple (*T, S*), where:

(1) $T = \{Task_i\}_{i=1}^{n}$ is the topological structure of workflow, and every task corresponds to a set of Web services with the certain functionality.

(2) $S = WS \cap SC$, $WS = \{ws_i\}_{i=1}^{n}$ is a set of Web services, and $SC = \{sc_i\}_{i=1}^{n}$ is a set of service clusters.

The Definition 3 introduces the concept Service Plan to specify the abstract structure of business workflow. The symbol S is a set of services and service clusters that may be selected to make up a composite service. There are four basic compositional structures in workflow: Sequence, Choice, Parallel and Iteration. Every complex workflow can be decomposed into these four structures. For simplicity and without loss of generality, we consider the sequence structure in the remainder of this paper.

Definition 4(QoS polarity and service comparison). There are two kinds of QoS, positive QoS and negative QoS. The service with bigger positive QoS is better than the one with smaller positive QoS, while the service with bigger negative QoS is worse than the one with smaller negative QoS. We use the partial order symbol "$\prec$" and "$\succ$" to express "worse than" and "better than" to unify the relation between different services. For example, the reliability of positive QoS, s1 $\prec$ s2 means s1.qos < s2.qos. The execution time of negative QoS, s1 $\succ$ s2 means s1.qos < s2.qos.

Definition 5(QoS-aware service selection with service clusters): Given a service plan *sp* with n tasks, QoS-aware service composition with service clusters is to select services from candidate services or service clusters. The aim is to complete the tasks as the service plan *sp* described in its topological structure with the maximum positive QoS value (reliability, reputation, etc.) or minimum negative QoS value (execution time, cost, etc.).

IV. Service Selection Algorithm

Given a service plan with n tasks in sequence topological structure. Let Q_i be the best QoS of service which is selected to complete task i, $Q^*_{i,j}$ be the best QoS of service cluster selected to complete workflow fragment from task i to task j. When atomic service with the simplest service cluster, we can get $Q^*_{i,j} = Q_i$ if i=j. Then the best global QoS of workflow fragment from task i to task j, $Q_i^j = \sum_{i=1}^{w} Q^*_{k_i,k_i+m_i}$ with constraints:

1) $k_v + m_v + 1 = k_{v+1} (\forall v \in [1, w])$

2) $k_1 = i$

3) $k_w + m_w = j$

4) $w + \sum m_i = j - i + 1$.

Creating service composition with best global QoS is the research goal in our problem. The objective is to maximize Q_1^n by adjusting the parameters. It is a parameters partition problem:

$$(w, m_1, \dots, m_w) = \arg\max Q_1^n$$

It is hard to solve the problem, but we can handle it step by step by dynamic programming method with the recursive relation:

$$Q_i^j = \max(Q_i^i + Q_{i+1}^j, \dots, Q_i^{j-1} + Q_j^j, Q^*_{i,j})$$

For example, $Q_1^1, Q_2^2, Q^*_{1,2}$ can be easily computed by comparing QoS of atomic services and service clusters separately, then $Q_1^2 = \max(Q_1^1 + Q_2^2, Q^*_{1,2})$. The pseudo code is shown in Table II.

TABLE. II
SERVICE SELECTION ALGORITHM

Workflow: 1->2->…->n

```
1:  //Initialize Q[i,j] with best QoS of atomic services or service clusters
2:  for i = 1 to n
3:    for j = 1 to n
4:      Q[i,j] = 0
5:      if i == j
6:        Q[i,j] =max{s.qos | s∈WS}
7:      else
8:        Q[i,j]=max {sc.qos |sc∈SC,
                   sc.funcs[first] = i ,
                   sc.funcs[end] = j}
9:  //recursive step
10: step = 1
11: while step <= n -1
12:   for i = 1 to n – step
13:     j = i + step
14:     Q[i,j] = max{Q[i,i]+Q[i+1,j],...,Q[i,j-1]
                  +Q[j,j],Q[i,j]}
15:   print constituent of selected Q[i,j]
16:   step = step + 1
17: print Q[1,n]
```

The algorithm is initialized as the traditional method (Line 1-8). With the recursive formula (Line 14), we build a bottom-up iteration to solve this problem (Line 10-16). The time complexity is $O(N^3)$.

V. EXPERIMENT AND ANALYSIS

In this section, a series of experiments are conducted on simulated data to evaluate our approach and explore how much it can improve the effectiveness and efficiency of service composition than the traditional method.

A. Effectiveness Evaluation

In this part, we just compare the different results of traditional approach without considering service clusters and our method, and investigate the factors which may improve the global QoS to create a better composite service.

1) Impact of task number

To examine the impact of task number, we vary the task number of end-user's workflow with parameters including candidate service and service cluster number. The average QoS of service and service cluster is unchanged. We take care of the global QoS of the created service composition. The experiments are repeated 100 times to get stable and accurate average results.

The results are shown in Fig.4. It is obvious that the global QoS of composite service generated by our method is better. The result of traditional method is proportional to the result of our method approximately. We next introduce a parameter called improvement coefficient (IC) which is defined as follow.

$$IC = \sum(sc.qos - \sum s.qos)/(|sc.funcs| \times |SC|), (sc \in SC, s \in sc.funcs)$$

It can be figured out from the experiment data that the scaling factor $\frac{QoS_{new}}{QoS_{old}}$ is equal to IC+1 which reveals how service cluster improve the average QoS of atomic service.

2) Impact of Candidate Service Number

There are lots of services and service clusters that can achieve the same task. The number of services or service clusters is called candidate service number. Two groups of experiments are conducted separately in different methods to show the impact factor.

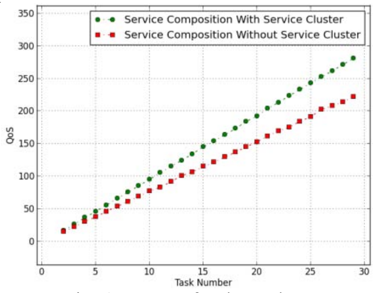

Fig. 4 Impact of task number

a) Candidate Service Number

As Fig.5 shown, when candidate service number is increased, the global QoS increases as well in these two methods because the increasing number makes it possible for the system to select better services. The improvement is much more significant that comparing to our method the traditional method only uses atomic services as candidate.

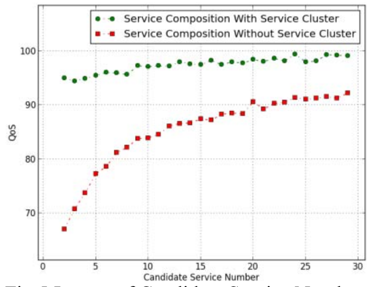

Fig.5 Impact of Candidate Service Number

b) Candidate Service Cluster Number

In Fig. 6, the increasing of service cluster number almost has nothing to do with performance of traditional method, because it never use service clusters as its candidate. But the increasing helps our method a lot, which has chances of finding better service clusters.

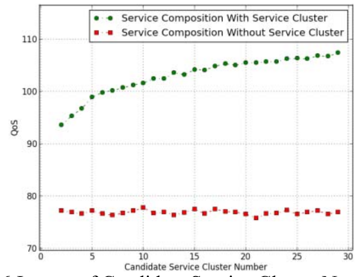

Fig.6 Impact of Candidate Service Cluster Number

B. Efficiency Evaluation

This part is to examine the execution performance of different methods. Since the execution time is short, we repeat each of them 300 times, and use the average execution time as actual execution time.

1) Impact of Task Number

Fig.7 shows that our method spends more time to deal with service composition. The time complexity is $O(N^3)$. It can be further reduced to $O(N^2)$ by preprocessing with min-heap, where the cost of min-heap (max-heap) in extremum searching is O(1).

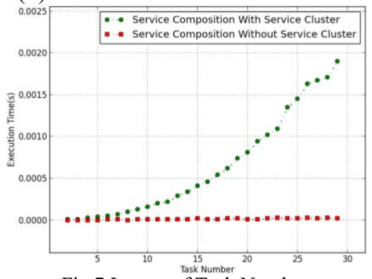

Fig.7 Impact of Task Number

2) *Impact of Candidate Service Number*

a) *Candidate Service Number*

Fig. 8 shows how the execution time increases by candidate service number. There is a fluctuation in diagram because the execution time is so short that there may be deviation during measuring. Both of methods spend time in selecting the best single service, which makes the time cost increased spontaneously.

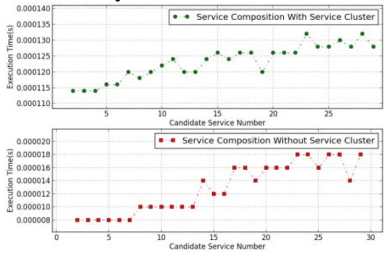

Fig.8 Impact of Candidate Service Number

b) *Candidate Service Cluster Number*

Fig. 9 shows that the execution time of traditional method is a relative fixed value. While the execution time of our method increases by the candidate service cluster number. That is because the traditional method only check the single service, but our method will check every possible service and service cluster.

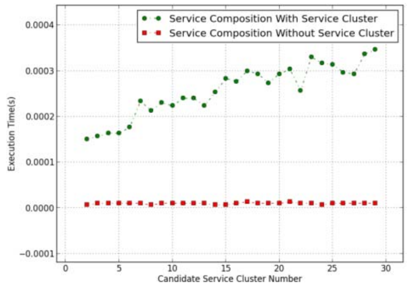

Fig.9 Impact of Candidate Service Cluster Number

VI. CONCLUSION AND FUTURE WORK

This paper introduces a new concept of service cluster which is applicable to a fragment of workflow. How to improve the attraction of cloud platform and the performance of service composition are discussed. Moreover, we give an algorithm to solve the service selection problem with service cluster and evaluate the performance of our proposed method by experiments. In future, the cost price to create service cluster in cloud platform will be taken into consideration. It is practical to build a dynamic mechanism to analyze and manage them.

ACKNOWLEDGEMENT

This research is supported in part by the National Natural Science Foundation of China under Grant No. 61170033 and the National High Tech Research and Development Plan of China under Grant No. 2014BAD10B02.

REFERENCES

[1] Z. Zheng, Y. Zhang, Lyu, M.R., Investigating QoS of Real-World Web Services, IEEE Transactions on Services Computing, 7(1):32-39, 2014

[2] M. P. Papazoglou et al., Service-oriented computing: A research roadmap, Int. J. Coop. Info. Syst., 17(2): 223–255, 2008.

[3] S. Deng, L. Huang, W. Tan, and Z. Wu. Top-k automatic service composition: A parallel framework for large-scale service sets". IEEE Transactions on Automation Science and Engineering. 11(3):891-905, 2014.

[4] S. Deng, L. Huang, D. Hu, J. Zhao, Z. Wu. Mobility-enabled Service Selection for Composite Services. IEEE Transactions on Services Computing. DOI 10.1109/TSC.2014.236579

[5] S. Deng, B. Wu, Z. Wu. Efficient planning for top-k Web service composition. Knowledge and Information Systems: An International Journal. 36(3):579-605, 2013.

[6] T. Yu, Y. Zhang, and K.-J. Lin. Efficient algorithms for Web services selection with end-to-end QoS constraints. ACM Transactions on the Web.1 (1), Article 6, 2007.

[7] J. El Haddad, M. Manouvrier, G. Ramirez, and M. Rukoz. QoS-driven selection of Web services for transactional composition. IEEE International Conference on Web Services, p653-660, 2008

[8] P. Xiong, Y. Fan, and M. Zhou.QoS-aware Web service configuration. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans.38 (4): 888-895, 2008.

[9] P. Bartalos and M. Bieliková. Qos aware semantic Web service composition approach considering pre/postconditions, IEEE International Conference on Web Services, p345-352, 2010.

[10] D. Chiu, S. Deshpande, G. Agrawal, and R. Li. A dynamic approach toward QoS-aware service workflow composition. IEEE International Conference on Web Services, p655-662, 2009.

[11] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. A lightweight approach for QoS-aware service composition. International conference on service oriented computing, Short Paper, 2004.

[12] S. Hwang, H. Wang, J. Tang, and J. Srivastava. A probabilistic approach to modeling and estimating the QoS of Web-services-based workflows. Information Sciences. 177(23):5484-5503, 2007

[13] M. Dumas, L. García-Bañuelos, A. Polyvyanyy, Y. Yang, and L. Zhang, Aggregate quality of service computation for composite services. International conference on Service-Oriented Computing, p213-227, 2010

[14] S. Uludag, K.-S. Lui, K. Nahrstedt, and G. Brewster. Analysis of Topology Aggregation techniques for QoS routing. ACM Computing Surveys. 39(3), Article 7, 2007.

[15] S. Deng, H. Wu, D. Hu, J. Leon Zhao. Service Selection for Composition with QoS Correlations. IEEE Transactions on Services Computing. DOI 10.1109/TSC.2014.2361138