International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.6 (2014), pp.171-182
http://dx.doi.org/10.14257/ijmue.2014.9.6.17

MultiGranular: An effective Service Composition Infrastructure for Multi-tenant Service Composition

Huihui Cai and Lizhen Cui*

School of Computer Science and Technology Shandong University, Jinan, 250101, China

chh_121@126.com, clz@sdu.edu.cn

Abstract

As a common delivery model in cloud computing, SaaS applications are becoming increasingly popular. Multi-tenancy is a key characteristics of SaaS applications. Service composition plays a main role in SaaS applications because of frequent composability and reusability of software services. With the development of cloud computing, one side, there are an increasing number of available services, other side, tenant' individual and diverse requirement become more intense, which makes service composition tend to be more complex. This paper proposes an effective infrastructure for service composition, MultiGranular, which provides a semantic basis for multi-tenant service composition. The MultiGranular supports the characteristic of hierarchy, uncertain correlation, inheritance and versioning, effectively responding to the requirements of complexity of service composition for cloud computing SaaS applications. The computing process of MultiGranular includes three steps: basic service clustering, correlation mining and hierarchy building. MultiGranular makes it possible to change disorder services into the hierarchy and ordered clustering services, and makes it easy to develop multi-tenant SaaS applications. The experimental results demonstrate that by utilizing the proposed method, complexity of multi-tenant service composition are solved and the efficiency of service composition has been improved greatly.

Keywords: *cloud computing, multi-tenant, service composition, hierarchy, uncertain correlation, inheritance and versioning*

1. Introduction

As the development of cloud computing and service computing, SaaS (Software as a Service) is widely used for bringing benefits to both software service providers and tenants. Service composition has become the dominant paradigm that utilizes services as fundamental elements for developing SaaS applications efficiently. However, one side, there is an increasing number of available services in large enterprise or domain, other side, tenant's individual and diverse requirement become more intense, which makes service composition tend to be more complex. In cloud computing, the complexity of multi-tenant service composition reflects in the following four aspects:

Multi-granularity. Business requirements are variant, and some are abstract and some are specific. Requirements show characteristic of multi-granularity. For example, someone submits his requirement as "travel plan" while others may say they want "book flight". How to satisfy different users' multi-granularity requirements is always an urgently solved problem.

ISSN: 1975-0080 IJMUE

Complexity of correlation. In the cloud SaaS model, services are sharing and isolation. Correlation relationships between services are more complex. In addition, the uncertainty of correlation relationships between services tends to be significantly increased.

Massive individuation. In the cloud environment, almost every tenant has special requirements for services in terms of UI, business process, data and so on. It indicates characteristic of individuation which varies a lot.

Frequency of evolution. Services always frequently evolve from one version to another version due to the change in business requirements. So it is necessary to effectively manage these services before and after evolution.

2. Related work

Over the last several years, different composition strategies have been proposed. [1] proposed an automated service composition approach based on service community. Task templates and service pools are included in the community to facilitate service discovery and composition. However, it is not possible to predefine all the templates. [2] proposed an uncertain QoS-aware skyline service selection method based on cloud model. With this method, the reliability and real-time performance of service selection have been guaranteed. [3] proposed a network-aware service composition method which considers the QoS of the network except the QoS of the service itself. It efficiently computes compositions with much lower latency. [4] built an extended quick service query list using WordNet and multiple heterogeneous ontologies to record service information and their associated semantic concepts towards more realistic service composition. The method alleviates users or developers from the burden of using complicated semantic service composition and can select and compose services to construct plans automatically. [5] proposed a multiobjective optimization framework for SLA-aware service composition leveraging a multiobjective genetic algorithm. [6] proposed an extended graph-planning based Top-K service composition method. By using service indexes and auxiliary nodes in extended planning graph, Top-K composition service can be found through one search.[7] put forward a matching algorithm SMA (Service Matching Algorithm) between cloud computing services of multiple input/output parameters, which considers the semantic similarity of concepts in parameters based on WordNet. [8] proposed a framework of service composition in multi-cloud base environments and three different cloud combination methods considering finding feasible composition sequence that contains minimum clouds. [9] proposed a QoS-based selection of multi-granularity web services for the composition, which allows services of multi-granularity available for selection. [10] proposed a service composition optimizing method which was user-oriented and based on the QoS value. Based on the fact that users usually put forth a coarse-granularity request. [11] proposed an on-demand service organization and recommendation method based on role, goal, process and service model for a domain problem to satisfy user's personalized and diverse requirements. [12] proposed an event-based web service composition in order to satisfy user's multiple requirements and be realized easily.

Though these methods above can achieve good results to some extent, they have not taken multi-tenant into consideration. Multi-tenancy is a key characteristics of SaaS applications, and there are some issues that traditional service composition approaches have not yet well addressed like multi-granularity requirements, complexity of correlation, massive individuation and frequency of evolution. With the lessons and

experiences from the traditional service composition approaches, we present an effective infrastructure for multi-tenant service composition.

3. Overview

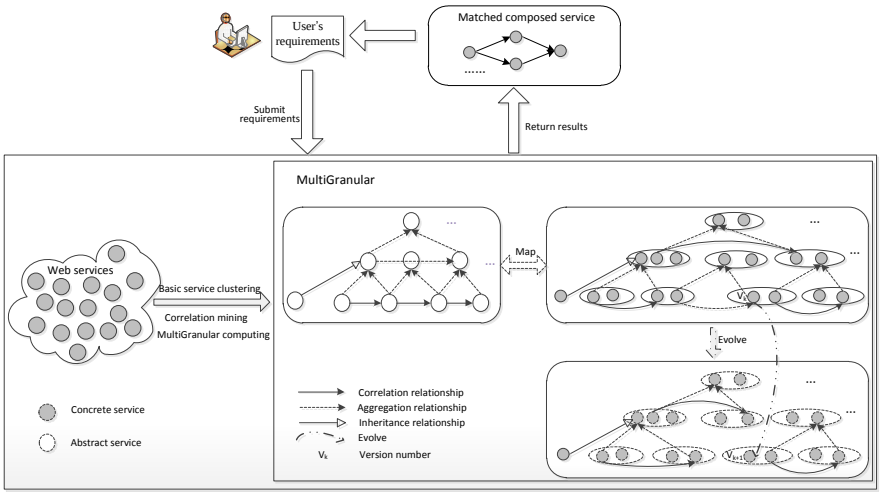

Figure 1. The Architecture of MultiGranular

The architecture of MultiGranular is shown in Figure 1. In order to respond to the complexity of multi-tenant service composition in cloud computing, we give the corresponding solving strategies:

Hierarchy. Considering from similar, correlation and aggregation relationships of the granularity among services, the process of MultiGranular computing is introduced, which includes basic service clustering, correlation mining and hierarchy building, which makes volumes of disorder services into the hierarchy and ordered clustering services, effectively responding to multi-granularity service composition requirements.

Uncertain correlation. Uncertain of correlation relationships are measured from the aspect of correlation degree between services. The computing of correlation degree is based on matching between concepts of services message and similarity measurement.

Inheritance. Tenants can firstly build sub service reflecting their individual requirement by inheriting on the mostly similar service with their requirements with least inheritance operation. Then corresponding concrete service will be got from which the inherited service maps to for customization. Inheritance is realized by modifying attributes of services, such as UI, business process, descriptive definition and data schema to satisfy tenants' individual requirements. In section 5 we will introduce some inheritance operations.

Versioning. The frequent evolution of services to meet business requirements changed always leads to many versions of the same original service. Therefore, it is necessary to realize version management mechanism.

4. MultiGranular Computing

As Figure 1 shows, MultiGranular includes two hierarchical models, in which abstract services maps to concrete services mutually. Formally, it is represented as a quintuple, (S, R_s ,R_c, R_a, R_i, De), where:

Definition 1. Service $S^{(v)}$= (UI, Business process, descriptive definition, data)

UI provides the uniform entrance for accessing service and describes the functional module and layout. Business process is the specific process for realizing particular goal and includes atomic process and composite process. It is consisted of services and flow relationships between them. Descriptive definition mainly declares the functionalities, such as input and output message and their data types. Data schema is consisted of a set of data items. v is the version number.

Definition 2. Similar relationship R_s. Services that can realize similar functionality because of similar semantic concepts are similar. Similar relationship between services is defined as

$$S_i \approx S_j, \quad Similar(S_i, S_j) \geq \sigma$$

There are many methods to compute similarity degree between services. In this paper, we compute the similarity degree based on their distance in the ontology.

$$Similar(S_i, S_j) = \frac{1}{dis(c_i, c_j)}, \qquad Similar(Input_i, Input_j) = \frac{1}{2} \times \left(\frac{\sum_{i=1}^{m} Max_{j=1}^{n} Similar(in_i, in_j)}{m} + \frac{\sum_{j=1}^{n} Max_{i=1}^{m} Similar(in_j, in_i)}{n} \right)$$

$$Similar(Output_i, Output_j) = \frac{1}{2} \times \left(\frac{\sum_{i=1}^{s} Max_{j=1}^{t} Similar(Out_i, Out_j)}{s} + \frac{\sum_{j=1}^{t} Max_{i=1}^{s} Similar(Out_j, Out_i)}{t} \right)$$

$$Similar(S_i, S_j) = w_1 \times Similar(Input_i, Input_j) + w_2 \times Similar(Output_i, Output_j)$$

In the MultiGranular, we firstly divide service set into several groups, services in each of which have relatively high similarity in functionality while services between different of which vary a lot. After basic services clustering, each service cluster can be represented as an abstract service through extracting the same concept respectively in the input and output message of services in the same cluster.

Definition 3. Correlation relationship R_c. Correlation relationship is the basis of service composition. According to the matching relationship between concepts of input and output message in the ontology, two forms of correlation relationships between services are defined as follows:

（1） $S_i \xrightarrow{R_c} S_j$, if $S_i.Output \supseteq S_j.Input, Similar(S_i.Output, S_j.Input) \geq \delta$

（2） $S_1 \cup S_2 \cup \cdots \cup S_n \xrightarrow{R_c} S_j$, if $S_1.Output \cup S_2.Output \cup \cdots \cup S_n.Output \supseteq S_j.Input,$ $\forall S_i, CD(S_i, S_j) = Similar(S_i.Output, S_j.Input) \geq \delta$

In the second step of building MultiGranular, the two forms of correlation relationship will be mined among abstract services according to semantic matching.

Definition 4. Aggregation relationship R_a. As coarse-grained services can hierarchically decompose to fine-grained services that have correlation relationship, we measure the two forms of aggregation relationship in the aspect that one service can realize composition function of services correlated.

(1) $S_i, S_j \xrightarrow{R_a} S_k$, if $S_i \xrightarrow{R_c} S_j$
$S_i.Input \approx S_k.Input$, namely $\forall S_i.Input, Similar(S_i.Input, S_k.Input) \geq \delta$
$S_i.Output \approx S_k.Output$, namely $\forall S_i.Output, Similar(S_i.Output, S_k.Output) \geq \delta$

(2) $S_1, S_2, \cdots, S_n, S_j \xrightarrow{R_c} S_k$, if
$S_1 \cup S_2 \cup \cdots \cup S_n \xrightarrow{R_c} S_j$
$S_1.Input \cup S_2.Input \cup \cdots \cup S_n.Input \approx S_k.Input, namely: \forall S_i, Similar(S_1.Input \cup S_2.Input \cup \cdots \cup S_n.Input, S_k.Input) \geq \delta$
$S_j.Output \approx S_k.Output, namely: \forall S_i, Similar(S_j.Output, S_k.Output) \geq \delta$

After the first two steps in MultiGranular building, correlation relationships among abstract services are mined. For correlated services, check whether there are coarse-grained services that can complete the composition functionality of them. Once found, aggregation relationships among them are built hierarchically. As each abstract service is mapped to the concrete service cluster, hierarchy building of service clusters will be correspondingly finished.

Definition 5. Inheritance relationship R_i. Services that have general attributes are called parent services; services that reflect individual requirement through inherited operations without changing logic realization are called child services.

Definition 6. $De: S \rightarrow G$ representing a decomposition of service $s \in S$ to a set of dags G. De(s) = (V, E) V=S, $E \subset \{R_c, R_a\}$, $\forall s_i, s_j \in S$, $(s, s_i) \in R_a$,$(s_i, s_j) \in R_c$.

5. Implementation of Individuation and Evolution

5.1. Inheritance

Inheritance can take place in several aspects of service, such as UI, business process, descriptive definition and data schema. Below is the corresponding inheritance operation for each attribute of service.

(1)For UI inheritance, there are two kinds of inheritance operation:

Change contents: For example, add new module or adjust font size.

Change structure: For example, adjust the layout of some modules or change the position of some images and menus.

(2)For business process inheritance, there are six kinds of inheritance operation:

Add business process: assume the original business process is BP_1= (i_1, S_1, o_1, F_1) and BP_2= (i_2, S_2, o_2, F_2) is the business process to be added to. Each business is represented as a quadruple: input, service set, output, and data flow relationship. Assume i and o are separately start service and end service in the business process. There are three kinds of inheritance operation:

Sequence adding: We call BP=(i, S, o, F) as the business process got through sequence adding BP_2 to BP_1,in which, $S=\{S_1 \backslash o_1\} \cup S_2$, $F=\{(s_i, s_j) \in F1 | s_j \neq o_1\} \cup \{(s_i,i_2) | (s_i,o_1) \in F_1\} \cup F_2$.

Parallel adding: We call BP=(i, S, o, F) as the business process got through parallel adding BP_2 to BP_1, in which $S=S_1 \cup S_2 \cup \{i, o\}$, $F=F_1 \cup F_2 \cup \{(i,i_1),(i,i_2),(o_1,o),(o_2,o)\}$.

Choice adding: We call BP=(i, S, o, F) as the business process got through choice adding BP_2 to BP_1, in which $S=S_1 \cup \{S_2 \backslash \{i_2,o_2\}\}$,$F=F_1 \cup \{(i_1,s_i)|(i_2,s_i) \in F_2\} \cup \{(s_i, s_j) \in F_2 \wedge s_i \neq i_2 \wedge s_j \neq o_2\} \cup \{(s_i,o_1)|(s_i,o_2) \in F_2\}$.

Delete business process: assume the original business process is BP= (i, S, o, F), BP_1 is the business process to be deleted; BP_2 is the business process that has parallel or choice relationship with BP_1 in BP. There are three kinds of inheritance operation:

Sequence deleting：We call BP'= (i', S', o', F') is the business process got through sequence deleting BP_1 from BP, in which, $S'=S\backslash \{S_1\}$, $i'=i$, $o'=i_1$, $F'=F\backslash \{(s_i, i_1)\}\backslash F_1$.

Parallel deleting: We call BP'=(i',S', o', F') is the business process got through parallel deleting BP_1 from BP, in which, $S'=S\backslash\{S_1\}$, $i'=i$, $o'=o_2$,$F'=F\backslash\{(i,i_1), (o_1,o)\}\backslash F_1$.

Choice deleting: We call BP'=(i',S', o', F') is the business process got through choice deleting BP_1 from BP, in which, $S'=S\backslash\{S_1\}$, $i'=i$, $o'=o_2$, $F'=F\backslash\{(i,s_1)(s_1,o)\}\backslash F_1$.

(4)For descriptive definition inheritance, there are two kinds of inheritance operation:

Add parameter i_j or o_j: $I=I \cup \{i_j\}$ or $O=O \cup \{o_j\}$ given the condition the added parameter does not affect the functionality.

Delete i_j or o_j: $I=I\backslash \{i_j\}$ or $O=O\backslash \{o_j\}$ given the condition the added parameter does not affect the functionality.

(5) For data schema inheritance, there are three kinds of inheritance operation:

Add data item d_i: $DS=DS \cup \{d_i\}$, DS= (name, birthdate). For example, add a data item named *affiliation* in a registry service table schema.

Delete data item d_i: $DS=DS\backslash \{d_i\}$, delete the attribute description of d_i and the dependency between data items.

Change data item d_i: For example, you can rename or change the type of the data item.

5.2. Versioning

Service versioning is used to identify different versions of the same service. The way how a version is determined and which characteristics included in the service of the version is defined by the service developer. Particularly, attributes about the version such as the number, difference from the service of original version and the description must be recorded so that users are provided many choices when selecting versions. Once the logic realization of the service is changed, a new version of the same service is generated. By applying versioning technique to services, different users can have different versions of the same service, in which some are the latest available version of the service while a limited number of users lag one or more versions behind. Once the new version of the service is released, some users may migrate to the new version while some users may stay on the original version for incompatibility.

6. Experimentation

The approach proposed in the paper is applied to the social security service system development based on SaaS. Developers develop services according to the service specification and publish them in the system. Those massive disorderly services are then organized by the approach as an orderly structure. When developers or tenants need to develop new social security application, it can be achieved quickly through service

composition using the approach. The experiment system contains thousands of services and runs on a computer running win 7 with a 2.33 GHz Dual Core processor and 2 GB memory. From the point of view of functions needed to achieve, this system can be divided into two main modules: (1) the building of MultiGranular; (2) the service composition system. As the first module has been previously shown, we focus on introducing the service composition system here.

6.1. Service Composition System

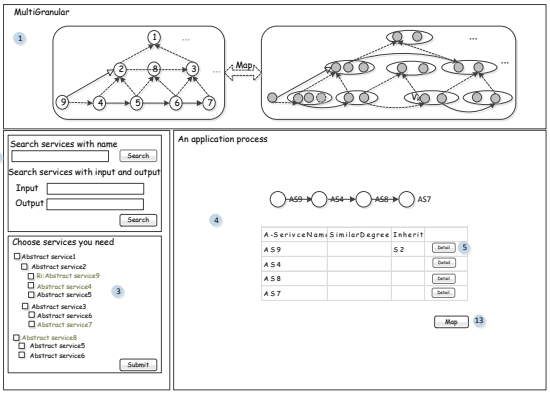

Figure 2. Service Composition Environment

As shown in Figure 2, when tenants login in to the system, MultiGranular will be displayed to them in graphical form (①). In the lower left of the interface, there are different categories of services, which cover the business functionality scope. Tenants can select services they need through clicking checkboxes aside by the abstract services or get services they need by clicking the search button after inputting the input parameters(②) of services they need as shown in the green service(③). Once the Submit button is clicked, service composition plans will be shown in the right area. Each abstract service in the plan is represented as a table, and each row is associated to a member service (④). The specified information of concrete services it maps can be checked by clicking Detail button (⑤).

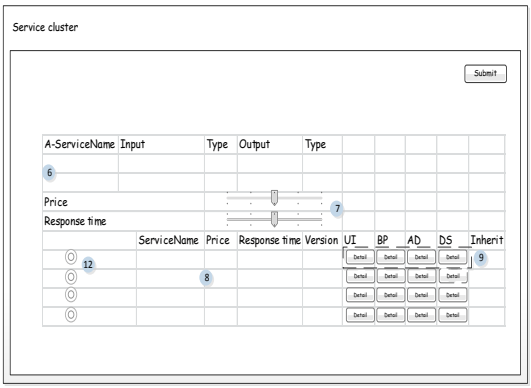

Figure 3. Service Cluster

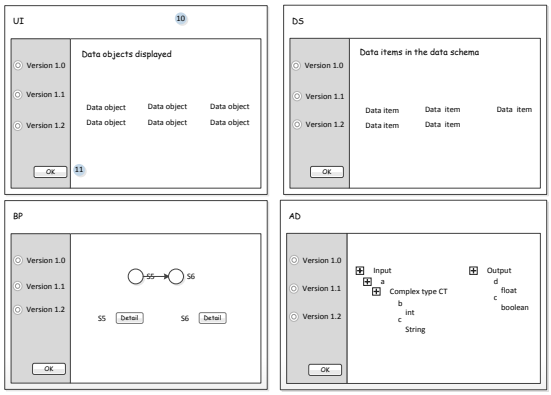

Figure 4. Service Attributes

As shown in Figure 3, each concrete service with a set of attributes can be seen in the Service cluster interface (⑥). To submit the preferences, users can tune up the bar (⑦) to

select services that satisfy QoS requirements (⑧). As shown in Figure 4, a set of attributes (UI, BP, AD, DS) can be checked (⑨~⑩). Then concrete services can be determined to be mapped after users checking attributes of services of different versions (⑪~⑫).

As shown in Figure 5, when all the concrete services of the abstract services are determined, the concrete services can be displayed when clicking the Map button (⑬~⑭). Then users can customize services they need by modifying attributes (⑮~⑰).Then the customization information and the services will be updated in the model. The frequent evolution of services to meet business requirements changed always leads to many versions of the same original service stored in the system (⑯).

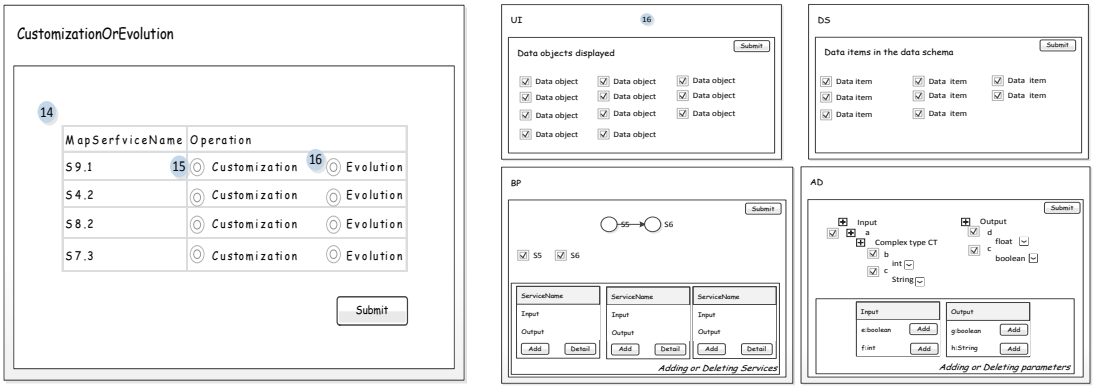

Figure 5. Service Customization

6.2. Service Granularity Analysis

To test the effectiveness of granularity on the performance of multi-tenant service composition, we firstly use several data sets to do the experiment. For each data set, we have manually created five requirements which can be achieved by services of different granularities ranged from 2 to 10. We take the average response time of those requirements as the final composition time.

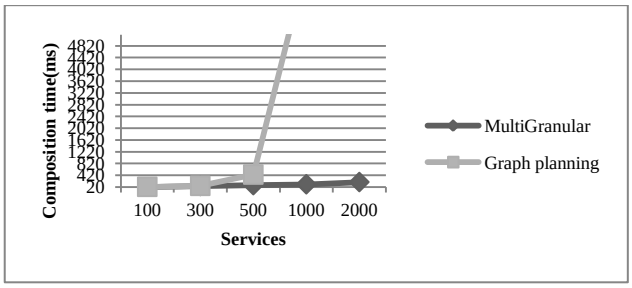

Figure 6. Service Composition Time

Figure 6 shows the results about the composition time for the specified service requirements in services ranged from 100 to 2000. It can be easily concluded that when the number of services is limited to be small, the time taken by the graph planning approach is nearly the same with the MultiGranular-based approach. However, as the number of services grows, graph planning approach takes far more time to find out the composition plans than the MultiGranular-based approach because it involved much semantic-related computing and

backtracking. On the contrary, the response time trend for MultiGranular-based approach is stable and does not increase much.

Secondly, we use 1000 services and manually created 5 requirements which can be achieved by services of different granularities ranged from10 to 14. Thus, we take the average response time of those requirements as the final composition time. Besides, because of the similarity of semantic concept, users can be provided with many service composition plans. Though those plans can satisfy functionalities needed, the correlation degree will be different. In order to test if our approach can achieve those plans on the basic of graph planning approach as we know it is sure to get suitable ones, we take the correlation degree of service composition plans as a measurement of accuracy. So we test the effectiveness in the view of correlation degree of service composition plans (SCPCD). The formula to compute correlation degree is as follows:

$$SCPCD=Similar(Req.Input, S_1.Input)\times\prod_{j=1}^{n-1} CD(S_j, S_{j+1})\times Similar(S_n.Output, Req.Output)$$

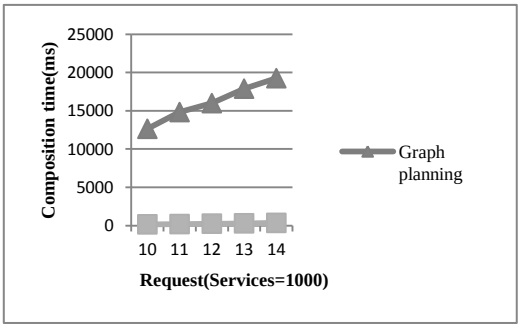

Figure 7.Service Composition Time

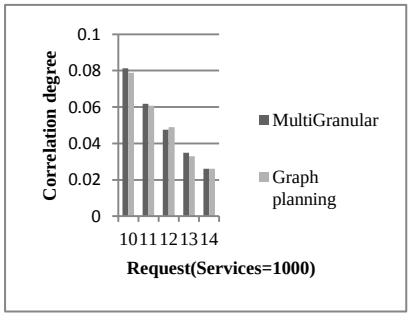

Figure 8. Correlation Degree of Service Composition Plans

Figure 7 shows the results about the composition time for the specified service requirements in services 1000. It can be easily concluded that the response time trend for MultiGranular-based approach is stable and does not increase much while the response time trend for the graph planning approach increases rapidly. As is shown in Figure 8, the MultiGranular-based approach can achieve service composition plans of similar correlation degree with the graph planning approach, which indicates that the MultiGranular-based approach can achieve correct service composition plans but in a short time.

6.3. Scalability Testing

The above experiment results of Figure 6, 7 and 8 were conducted under a small service number of services. To illustrate the scalability, we also devised 5 requirements which can be achieved by services of different granularities ranged from 2 to 10 in the service ranged from 3000 to 5000. As we can see in Table 1, when the service number becomes large, it is very hard for graph planning approach to retrieve the composition plans because the searching time blows up exponentially and memories quickly run out. On the contrary, the response time trend for MultiGranular-based approach does not increase much.

Table 1. Service Composition Time

	MultiGranular	Graph planning
3000	264ms	153340.7
4000	297.6ms	--
5000	310.2ms	--

7.4. Analysis of Basic Service Clustering and Correlation Mining

In order to illustrate the effectiveness of mining correlation relationship in the situation of clustering, we demonstrate the number of mined correlation relationships and the time for that with different service number ranged from 100 to 2000. Figure 9 shows that the time consumed after clustering is much less than that without clustering, and Figure 10 shows that the number of mined correlation relationships is almost the same compared to that without clustering. As many services are similar in functionality and are not correlated, they can be filtered through clustering for preprocessing.

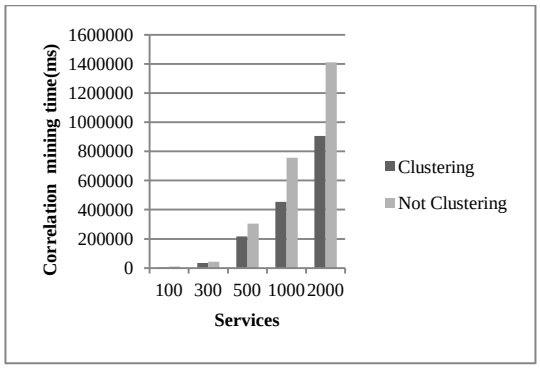

Figure 9. Correlation Mining Time

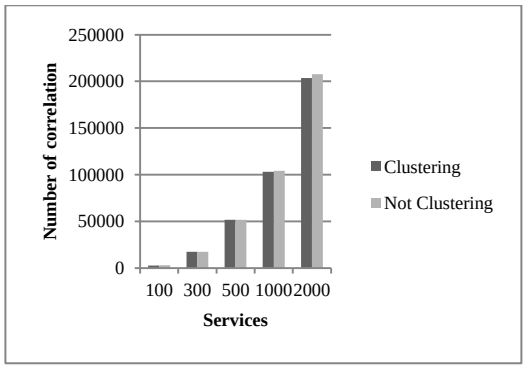

Figure 10. Number of Correlation

7. Conclusions and Future Work

Traditional semantic-based composition approaches have not yet well addressed several challenging issues for multi-tenant service composition in cloud computing. In this paper, we have presented an innovative composition technique by combining clustering and correlation mining together to build MultiGranular as a semantic basis for supporting multi-tenant service composition. Based on MultiGranular, the computational efficiency can be significantly improved while guaranteeing the individual and diverse requirement. And frequent service evolution has been also taken into consideration. Besides, our approach can be more scalable to a large scale of service sets and alleviate tenant or developers from the burden of using complicated semantic knowledge for service composition; thus make service composition toward more realistic and easier. An experiment and evaluation has been conducted to demonstrate the effectiveness of our proposed approaches. In future, we will turn our approach to the standardized ontologies on Internet to build more comprehensive MultiGranular. In addition, we will further improve the efficiency of the process of basic services clustering and correlation mining.

Acknowledgements

This work has been supported by the National Natural Science Foundation of China under Grant No. 61003253; the Natural Science Foundation of Shandong Province of China under Grant No. ZR2010FM031, ZR2010FQ010.

Reference

[1] X. Z. Liu and H. Mei, Science China (Information Sciences), vol. 53, no. 1, **(2010),** pp. 50-63.
[2] S.G. Wang, Q. B. Sun, G. W. Zhang, and F. C. Yang, "Journal of software", vol. 23, no. 6, **(2012),** pp. 1397-1412.
[3] A. Klein, F. Ishikawa, and S. Honiden, "Towards network-aware service composition in the Cloud", Proceedings of the 21st international conference on World Wide Web, **(2012)** April 16–20, Lyon, Japan.
[4] K. J. Ren, N. Xiao, and J. J. Chen, "IEEE Transaction on services computing", vol. 4, no. 3, **(2011),** pp. 216-229.
[5] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "IEEE Transaction on services computing", vol. 5, no. 3, **(2012),** pp. 358-372.
[6] M. Xu, L. Z. Cui, and Q. Z. Li, "Acta Electronica Sinica", vol. 40, no. 7, **(2012),** pp. 1404-1409.
[7] C. Zeng, X. Guo, W. J. Ou, and D. Han, "Cloud computing service composition and search based on semantic", Proceedings of the 1st International Conference on Cloud Computing, **(2009)** December 1-4; Berlin, Germany.
[8] G. B. Zou, Y. X. Chen, Y. Xiang, R. Y. Huang, and Y. Xu, "AI planning and combinatorial optimization for web service composition in cloud computing", Proceedings of the International Conference on Cloud Computing and Virtualization, (**2010)** May 17–18, Singapore.
[9] B. Zhou, K. Yin, Y. Jiang, H. H., and S. Zhang," Journal of software", vol. 6, no. 3, **(2011)**, pp. 366-373.
[10] J. W. Ma, D. K. Guo, J. X. Liu, and X. S. Luo, "Journal of Harbin Engineering University", vol. 31, no. 10, **(2010),** pp. 1360-1366.
[11] J. X. Liu, K. Q. He, J. Wang, D. H. Yu, Z.W. Feng, D. Ning, and X. W. Zhang, Chinese Journal of Computers, vol. 36, no. 2, **(2013),** pp. 238-251.
[12] X. Li, B. Cheng, G.W. Yang, and Q.H. Liu, "Journal of Software", vol. 20, no. 12, **(2009)**, pp. 3101-3116.

Authors

Huihui Cai, she is a M.S. candidate in the School of Computer Science and Technology at the Shandong University. Her research interest is service computing.

Lizhen Cui, he received the PhD degree in computer science from the Shandong University in 2006.He is an associate professor in the School of Computer Science and Technology at Shandong University. His research interests include service computing, workflow and data management for cloud computing