Towards Efficient Service Composition in Multi-Cloud Environment

Junwen Lu
College of computer and Information Engineering
Xiamen University of Technology
Xiamen, China

Yongsheng Hao
Information management department
Nanjing University of Information Science & Technology
Nanjing, China

Lina Wang
School of Electronic and Information Engineering
Nanjing University of Information Science & Technology
Nanjing, China

Mai Zheng
Department of Computer Science
New Mexico State University
Las Cruces, NM, USA

***Abstract*-- In this paper, we propose a theoretical model for the service composition in MCE. A user sends service requests to the MCE. Each service request can be satisfied from multiple clouds (i.e., service composition). Given this model, we then design a multi-layer algorithm to minimize the service composition overhead. The overhead is measured through two fundamental metrics: (1) the average number of clouds (ANC) involved in the service composition, and (2) the average number of service files (ANS) examined. While simple, the two metrics capture the fundamental communication overhead across clouds and within clouds, respectively. Preliminary evaluation based on simulation show that our algorithm outperforms the previous approaches in terms of both ANC and ANS**.

Keywords: ***cloud computing; multi-cloud environment; service composition*; *data sharing***

I. INTRODUCTION

Cloud computing is transforming the IT industry. More and more service can be get from the Cloud platform. How to combine the service from multiple Clouds become a problem in the Cloud resource management.

More generally, our contributions include: (1) a theoretical model for the service composition problem in multi-cloud environment which captures two fundamental metrics of communication overhead; (2) a multi-layer algorithm to minimize the average number of Clouds under the given model; (3) simulation results on the effectiveness of the proposed model and algorithm. The remainder of this paper is organized as follows: Section 2 discusses the related work. Section 3 introduces the data sharing model. In Section 4, we propose a multi-layer algorithm for minimizing the communication overhead. The simulations and discussions are presented in Section 5. We conclude the paper and highlight the future research directions in Section 6.

II. RELATED WORK

Cloud computing has characteristics such as On-demand self-service, elasticity, broad network access and so on [1]. Those characteristics make Cloud service can be widely used in the Cloud. A Cloud service can provide one or more functionalities to users with restriction and rulers through the interface between the user and the Cloud service [2-6].

Different to the prior work, this paper focuses on reducing the data sharing overhead incurred during the service composition in the multi-cloud environments. We try to reduce the average number of the Clouds that which is used in the service composition. If the value is too large, it would enlarge the time of service composition and the file size that need to be transmitted between different Clouds, which would bring negative effort to the scheduling. There are many composition models, such as parallel split pattern, synchronization pattern, that is used in the true system. Most of time, a service composition request always includes some patterns of the composition models. Various composition models are often the same service composition problems. So, in this paper, we only focus on the sequential model. Others composition models can be transformed into the sequential model in practice. Note that a web service also involves many QoSs issues, such as price, reliability, availability and reputation [7-9], which are not the focus of this paper.

Some service composition methods have been proposed in a single cloud environment, which do not consider the sharing opportunities among different clouds and thus limit the ability of collaboration among clouds. Different from those methods, we try to solve the service composition in multi-cloud environments. First of all, we propose a simple model for the service composition problem in the multi-cloud environment.

III. SYSTEM MODEL

As shown in Fig. 1, the model consists of five main components:

- A MCE has multiple Clouds. $MCE = \{C_1, C_2, ..., C_{MCEN}\}$. Each Cloud also has a set of service files, Each service file also has multiple services. $len(S_i)$ returns the number of service candidates that in the service file.

Supported by Natural Science Foundation of Fujian（2013J05103）and Open Project of Key Laboratory of Jiangxi（NSS1403）(*sponsors*).

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.69

• The user submits the user request to the system. Cloud manager and Service manager works together to finish the service composition request.

• Every Cloud registers to the MCE Data Center, and reports the service files that the Cloud can be provided to the users.

• MCE Data Center records the condition of the every Cloud. It also checks the state of every service and changes the record with periodicity.

• Cloud manager gets the request from the user and gets the system data from the MCE Data Center. Based on those data, it decides which Cloud should be allocated to the user.

• Service manager gets the details of the Cloud from MCE Data Center. At the same time, it decides which service file should be selected and where (in which Cloud) the service file should be selected.

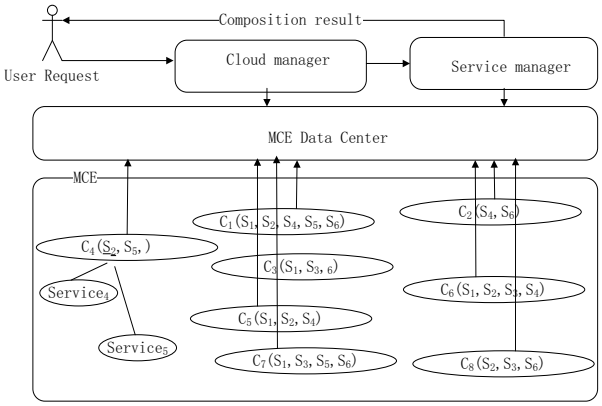

Fig. 1. The model of the service composition in MCE

There are two optimization targets under this model:

(1) to minimize the number of examined services, and

(2) to minimize the number of Clouds involved in the service composition sequence.

The number of examined services decides the time of finding the composition sequence. The number of Clouds that involved in the composition sequence decides the communication cost between different Clouds. The two parameters are the most important parameters in our system. Note that the two goals are similar to the ones used in the previous studies.

IV. SERVICE COMPOSITION ALGORITHMS

To solve the service composition problem under MCE, we propose our method, DC-Cloud, in this section. The composition method decides select which Clouds and which service files in each Cloud to satisfy the request of the user. The composition method is illustrated in Algorithm 1.

Algorithm 1: DC-Cloud: Service composition based on Data Center information

1 **Input**: Service composition request R and MCE information
2 **Output**: the selected Clouds and the selected service files in each selected Cloud
3 $MCE = \{C_1, C_2, ..., C_{MCEN}\}$; // There are MCEN Clouds in the MCE
4 $SF = \{S_1, S_2, ..., S_{SN}\}$; //There are SN service files totally
5 $SFN = \{SF_1, SF_2, ..., SF_{SN}\}$; // The number of atom services in every service file
6 $R = \{SFR_1, SFR_2, ..., SFR_{SFRN}\}$; //there are SFRN service files in the service composition request R, we sort service files in R in the descending order of the total number of the service files in the Clouds.
7 SFC_i is the set of the services in the Cloud C_i ;
8 $MCES_j^s = \{\bigcup C_t \mid S_j \in SFC_t\}$; // $MCES_j^s$ is the Cloud set that which has the service S_j ;
9 $SelC = \Phi$;//set the selected Cloud set to NULL
10 **For** every service file S_{stemp} in R
11 $SelC = SelC \cup MCES_{stemp}$
12 **EndFor**
13 **ScheduleEmpty( )**;
14 **Max-Min-Min( )**;

Lines 1~8 of Algorithm 1 is the initializations of the MCE. Those parameters are listed as follows:

- MCE is the multi-cloud environment, there are $MCEN$ Clouds in the MCE;

-There are SN kinds of service files totally and they are listed in SF , the number of each service is listed in SFN ;

- SFC_i is the set of services in the Cloud C_i ;

- $MCES_j$ is the Cloud set that which have the service S_j ;

- R is the service composition request;

- $SelC$ is the Cloud that at least has a same service candidate to the service composition request of R (Line 7-10);

ScheduleEmpty( ) is the first step of service composition method, which tries to composite service in the Cloud whose services are in the service composition request. Algorithm 2 gives the details.

Max-Min-Min( ) is the second step of the composition method, which tries to composite the rest services in other Clouds. Algorithm 3 is the details.

In Algorithm 2, first of all, we check the Cloud which has the service files are all in the service composition request R (Lines 1, $C - C \bigcap R == \Phi$). If there is more than one Cloud that can satisfy the condition, we select the Cloud which has a larger number of services (Lines 6~10), which could potentially reduce the number of Clouds involved in the service composition. If we can find the Cloud, we delete the selected Cloud and update the state information of the MCE (Lines 13~16).

Algorithm 2: ScheduleEmpty()

1 **While** (Exist a Cloud C in $SelC$ makes $C - C \bigcap R == \Phi$) **Do** {
2 **For** every Cloud C_{ctemp} in $SelC$
3 $ol = 0$; // records the number of the service in the selected Cloud
4 $selcloud = \Phi$; // records the selected Cloud
5 **If** $SFC_{ctemp} - SFC_{ctemp} \bigcap R == \Phi$
6 $l = Len(SFC_{ctemp})$;

```
7            If ( l > ol )
8                selcloud = SFC_ctemp ;
9                ol = l ;
10           EndIf
11       EndIf
12   Endfor
13   If selcloud != Φ
14       R = R − selcloud ; // selects the service in the selcloud
15       SelC = SelC − selcloud ; //Deletes selcloud from SelC ;
16   EndIf
17 }
```

In the following, we give three rules for the rest services (the second step，Algorithm 3):

(1) The Cloud which has the largest number of the same service files to R will be first selected (Max), so that we can reduce the number of Clouds involve in the service composition;

(2) If two Clouds has the same value in (1), we select the Cloud which has the minimum of the different service files to R (Min), so that we can reduce the resource fragment;

(3) If two Clouds has the same value in (1) and (2), we select the Cloud which has a minimum number of different service files to R in SFN (Min). This is also for reducing the resource fragment.

Line 2~5 (Algorithm 3, same in the following) are the initiation of the second step. $Len(SFC_{ctemp} \cap R)$ returns the number of the same services in SFC_{ctemp} and R. Lines 8~13 are the rule (1). Lines 15~21 are the rule (2). Line 15 gets the number of services that in SFC_{ctemp}, but not in R. Lines 22~32 are the rule (3). Line 24 gets the minimum number of the services in nst ($min1$) according to SF. Line 25 gets the minimum number of the services in $nservice$ ($min2$) according to SF. If $min1$ is less than $min2$, we would select SFC_{ctemp}.

Algorithm 3: Max-Min-Min

```
1  While R is not a NULL Do{
2      ol = 0; // records the number of the same services in the selected Cloud and R
3      selcloud = Φ ; // records the selected Cloud
4      nol = 0; // records the number of same services in the selected Cloud but not in R
5      nservice = Φ; // records the services in the selected Cloud but not in R
6      For every Cloud C_ctemp in SelC
7          l = Len(SFC_ctemp ∩ R)
8          If ( l > ol ) //executes service composition
9              selcloud = SFC_ctemp ;
10             ol = l ;
11             nol = Len(SFC_ctemp − R ∩ SFC_ctemp) ;
12             nservice = SFC_ctemp − R ∩ SFC_ctemp
13         EndIf
14         If ( l = ol & ol != 0 )
15             nolt = Len(SFC_ctemp − R ∩ SFC_ctemp) // the number of services that in SFC_ctemp , but not in R
16         If ( nolt < ol ) //executes service composition
17             selcloud = SFC_ctemp ;
18             ol = l ;
19             nol = Len(SFC_ctemp − R ∩ SFC_ctemp) ;
20             nservice = SFC_ctemp − R ∩ SFC_ctemp ;
21         EndIf
22         If ( nolt = ol )
23             nst = SFC_ctemp − R ∩ SFC_ctemp ;
24             Gets the minimum number of the service in nst ( min1 ) according to SF ;
25             Gets the minimum number of the service in nservice ( min2 ) according to SF ;
26             If ( min1 < min2 ) //executes service composition
27                 selcloud = SFC_ctemp ;
28                 ol = l ;
29                 nol = Len(SFC_ctemp − R ∩ SFC_ctemp) ;
30                 nservice = SFC_ctemp − R ∩ SFC_ctemp
31             EndIf
32         EndIf
33     EndIf
34 EndFor
35 }
```

To better understanding of our method, we will give an example of the method in Section V.A.

Suppose the average length of service files is l_{avg}, there are MCEN Clouds and SN service files. The complexity of our algorithm can be defined as follows:

$O(f) = O(\text{algorithm 1}) + O(\text{algorithm 2}) + O(\text{algorithm 3})$
$= O(MCEN*SN*l_{avg}) + O(MCEN*SN*l_{avg}) + O(MCEN*SN*l_{avg})$
$= MCEN*SN*l_{avg}$

V. EVALUATION

In this section, we evaluate our model and algorithms via simulation on public dataset. We compare our method, DC-Cloud, with All-Clouds [10], Base-Clouds [10], Smart-Clouds [10] and COM2 [10]. All-Clouds adds the Cloud one by one that which includes the service file(s) in the service composition request until it satisfies all services. Base-Clouds tries to find the minimum number of Clouds in the final service composition sequence. Smart-Clouds is an approximation algorithm which first models a multi-cloud environment as a tree and then finds a minimum request set by searching the tree. COM2 always selects the Cloud with the maximum number of services. DC Clouds first evaluate the number of the service and the service composition request, then according to the condition, Max-Min-Min is used to the service composition.

Simulation 1 compares those methods under the same environment that has been taken in [10]. Different to them, we evaluate them when there are more service composition requests. Simulation 2 give an evaluation under a new environment, where are multiple Clouds and multiple

service composition requests.

A. Simulation1

TABLE I is the number of Cloud services in each service file. TABLE II is the setting of five MCEs of our simulation. "NC" is the number of combined Clouds which decides the communication costs and financial charges between Clouds. "NS" is the number of the examined services which decides the time to find the composition sequence. We hope that both NC and NS have a smallest value, but most of time, the dropping of one value always induces the increasing of the other. "SUM" is the total value of the related parameters. We evaluate three service composition requests under the same environment. TABLE III, IV and V are the simulation results.

To All-Clouds, DC-Cloud always has a smaller value in NC. Though sometimes, COM2 has a lower value in NS, but COM2 has a larger value in NC, even the value is more than the value of Base-Clouds and Smart-Clouds.

In general, DC-Cloud has a small value in NC. At the same time, it also has a relatively small value in NS. So, DC-Cloud can find a composition sequence that not only in a short time, but also has a short communication cost between different Clouds.

B. Simulation 2

The simulation environment has five service composition requests and six service files. We compute the composition sequence one by one. When one is finished, we would delete those service files from the Cloud that have been assigned to a service composition request. The numbers of services in every service file are listed in TABLE VI and the five service composition requests are listed in TABLE VII. TABLE VIII lists the five multi-cloud environments. TABLE IX gives the details of the composition sequence for R_1=(S_1, S_3). (The service file with underline means the service file that has been selected)

a) Comparisons for the service composition requests R_1

In this section, we pay attention to the parameters for the service composition request R_1. Fig. 2 and Fig. 3 are the values of NC and NS in the five MCEs for the service composition request R_1. The X axis is the environment of the five MCEs. The Y axis of Fig. 2 is the value of NC and the Y axis of Fig. 3 is the value of NS. We find DC-Cloud always has the lowest value in NC and has a relatively lower value in NS. Fig. 4 and Fig. 5 are the average value of NS and NC of the service composition request R_1 of the five MCEs. The X axis is the five composition methods: All (All-Clouds), Base (Base-Clouds), Smart (Smart-Clouds), COM2 (COMposition), DC (DC-Cloud). The Y axis of Fig. 4 is the average value of NC and the Y axis of Fig. 5 is the average value of NS. From Fig. 4 and Fig. 5, we find that DC-Cloud has the lowest value in NS. At the same time, it also has a relatively lower value in NS. Only the value of NS of COM2 is less than the value of DC-Cloud, but the average value of NC of COM2 is more than DC-Cloud.

For the first service composition request R_1, we find the result has the same pattern with the simulation in Section V.A. To further evaluate our method, after we have finished R_1, we would finish the others service composition requests one by one.

TABLE I. THE NUMBER OF CLOUD SERVICES IN EACH FILE

Service File	S_1	S_2	S_3	S_4	S_5
Number of services	2	3	8	3	3

TABLE II. MCE SETTINGS FOR THE SIMULATION

	MCE_1	MCE_2	MCE_3	MCE_4	MCE_5
C_1	S_1, S_2, S_3	S_1, S_2	S_1, S_3, S_5	S_2, S_3, S_5	S_1, S_2
C_2	S_4, S_5	S_3	S_5	S_3, S_4	S_2, S_3
C_3	S_3, S_4	S_2, S_5	S_1, S_2	S_1, S_2, S_3	S_3
C_4	S_1, S_2, S_3, S_5	S_1, S_4, S_5	S_3, S_4	S_4, S_5	S_1, S_4, S_5

TABLE III. PERFORMANCE OF DIFFERENT METHODS FOR R_1=(S_1, S_2, S_3, S_4)

Algorithm	**All-Clouds**		**Base-Clouds**		**Smart-Clouds**		**COM2**		**DC-Cloud**	
Parameters	NC	NS	NC	NS	NC	NS	NC	NS	NC	NS
MCE_1	3	46	2	65	2	70	2	35	2	46
MCE_2	4	27	3	148	3	48	3	45	3	27
MCE_3	3	32	2	128	2	48	3	50	2	29
MCE_4	4	44	2	68	2	140	3	49	2	44
MCE_5	4	32	2	112	3	56	2	30	3	32
SUM	18	181	11	521	12	362	14	209	12	178

TABLE IV. PERFORMANCE OF DIFFERENT METHODS FOR R_2=(S_1, S_2, S_3)

Algorithm	**All-Clouds**		**Base-Clouds**		**Smart-Clouds**		**COM2**		**DC-Cloud**	
Parameters	NC	NS	NC	NS	NC	NS	NC	NS	NC	NS
MCE_1	2	29	1	13	1	13	1	16	1	13
MCE_2	4	27	2	50	2	40	3	27	2	27
MCE_3	1	13	2	66	2	54	2	18	2	29
MCE_4	1	13	1	38	1	38	2	27	1	38
MCE_5	3	24	2	48	2	51	3	24	2	48

SUM	11	105	8	215	8	196	11	112	8	155

TABLE V. PERFORMANCE OF DIFFERENT METHODS FOR R_3=(S_1, S_2)

Algorithm	**All-Clouds**		**Base-Clouds**		**Smart-Clouds**		**COM2**		**DC-Cloud**	
Parameters	NC	NS	NC	NS	NC	NS	NC	NS	NC	NS
MCE_1	1	29	1	13	1	84	1	1687	1	56
MCE_2	1	13	1	5	1	27	2	13	1	5
MCE_3	2	18	1	21	1	37	2	18	1	23
MCE_4	1	13	1	38	1	44	2	27	1	44
MCE_5	2	13	1	5	1	32	2	13	1	5
SUM	7	86	5	92	5	224	9	87	5	133

TABLE VI. THE NUMBER OF CLOUD SERVICES IN EACH SERVICE FILES

Service file (*S*)	***S*$_1$**	***S*$_2$**	***S*$_3$**	***S*$_4$**	***S*$_5$**	***S*$_6$**
Number of service (*S*)	5	4	3	2	7	6

TABLE VII. FIVE SERVICE COMPOSITION REQUESTS RE={R_1, R_2, R_3, R_4, R_5}

Request	***R*$_1$**	***R*$_2$**	***R*$_3$**	***R*$_4$**	***R*$_5$**
Service file	S_1, S_3	S_3, S_4, S_5, S_6	S_1, S_3, S_4, S_6	S_1, S_3, S_4, S_5, S_6	S_2, S_5, S_6

TABLE VIII. THE MULTI-CLOUD ENVIRONMENT SETTING FOR MCE_1~MCE_5

Cloud	***MCE*$_1$**	***MCE*$_2$**	***MCE*$_3$**	***MCE*$_4$**	***MCE*$_5$**
C_1	S_1,S_2,S_4,S_5,S_6	S_1,S_2,S_4,S_6	S_2,S_3,S_5,S_6	S_1,S_2,S_3,S_4, S_6	S_2,S_3,S_5,S_6
C_2	S_4,S_6	S_3,S_4,S_6	S_1,S_6	S_2,S_3, S_5,S_6	$\underline{S_1}$,$\underline{S_3}$
C_3	S_1,S_3,S_6	S_1,S_4,S_6	S_1,S_5,S_6	S_2	S_2,S_4,S_5
C_4	S_2,S_5	S_2,S_3,S_6	S_1,S_3,S_4	S_1,S_3,S_4,S_5,S_6	S_1,S_3,S_4,S_6
C_5	S_1,S_2,S_4	S_1,S_3,S_5	S_1,S_2,S_4	S_1,S_2,S_3,S_4,S_5,S_6	S_1,S_2,S_3,S_5,S_6
C_6	S_1,S_2,S_3,S_4	S_1,S_5,S_6	S_2,S_3,S_6	S_1,S_5	S_1,S_3,S_4,S_5,S_6
C_7	S_1,S_3,S_5,S_6	S_2,S_3,S_4,S_5,S_6	S_1,S_4,S_5,S_6	S_4,S_5,S_6	S_2,S_5,S_6
C_8	S_2,S_3,S_6	S_2,S_3,S_6	S_1,S_3,S_4, S_6	S_2,S_5	S_1,S_6

TABLE IX. DC CLOUD FOR THE COMPOSITION SERVICE OF R_1（S_1, S_3）

Cloud	***MCE*$_1$**	***MCE*$_2$**	***MCE*$_3$**	***MCE*$_4$**	***MCE*$_5$**
C_1	S_1,S_2,S_4,S_5,S_6	S_1,S_2,S_4,S_6	S_2,S_3,S_5,S_6	$\underline{S_1}$,S_2,$\underline{S_3}$,S_4, S_6	S_2,S_3,S_5,S_6
C_2	S_4,S_6	S_3,S_4,S_6	S_1,S_6	S_2,S_3,S_5,S_6	$\underline{S_1}$,$\underline{S_3}$
C_3	$\underline{S_1}$,$\underline{S_3}$,S_6	S_1,S_4,S_6	S_1,S_5,S_6	S_2	S_2,S_4,S_5
C_4	S_2,S_5	S_2,S_3,S_6	$\underline{S_1}$,$\underline{S_3}$,S_4	S_1,S_3,S_4,S_5,S_6	S_1,S_3,S_4,S_6
C_5	S_1,S_2,S_4	$\underline{S_1}$,$\underline{S_3}$,S_5	S_1,S_2,S_4	S_1,S_2,S_3,S_4,S_5,S_6	S_1,S_2,S_3,S_5,S_6
C_6	S_1,S_2,S_3,S_4	S_1,S_5,S_6	S_2,S_3,S_6	S_1,S_5	S_1,S_3,S_4,S_5,S_6
C_7	S_1,S_3,S_5,S_6	S_2,S_3,S_4,S_5,S_6	S_1,S_4,S_5,S_6	S_4,S_5,S_6	S_2,S_5,S_6
C_8	S_2,S_3,S_6	S_2,S_3,S_6	S_1,S_3,S_4, S_6	S_2,S_5	S_1,S_6

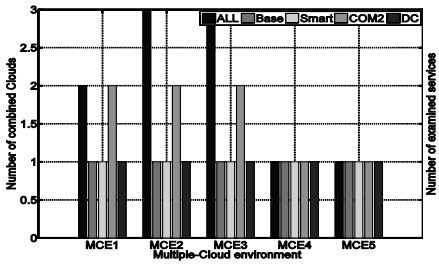

Fig. 2. NC of R_1

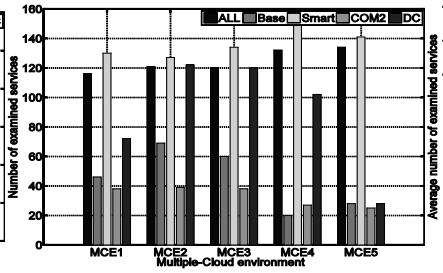

Fig. 3. NS of R_1

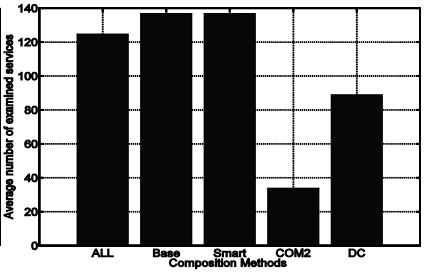

Fig. 4. Average NS of R_1

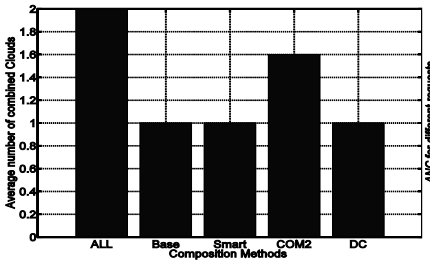

Fig. 5. Average NC of R_1

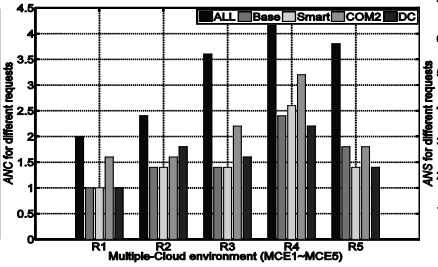

Fig. 6. ANC of all the service composition requests

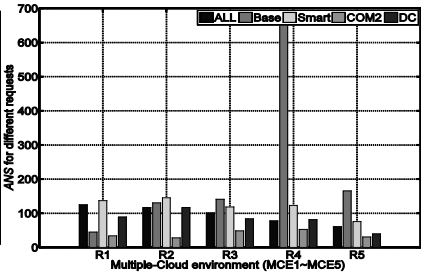

Fig. 7. ANS of all service composition requests

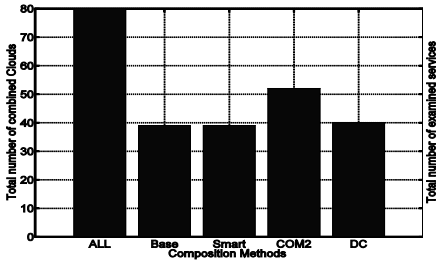

Fig. 8. Total NC of all the service

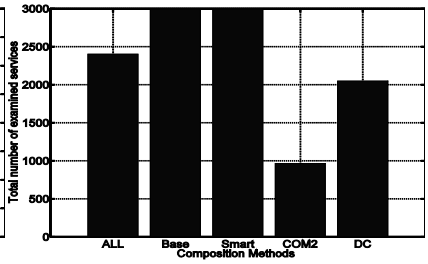

Fig. 9. Total NS of all the service composition requests

b) Comparisons for all service composition requests

Fig. 6 and Fig. 7 are the average value of NC and NS under the five MCEs. The X axis is the five service composition requests from TABLE VII. The Y axis in Fig. 6 is the average value of NC (ANC) of the five service composition requests and the Y axis in Fig. 7 is the average value of NS (ANS) of them. All-Clouds always has the largest value in ANC and COM2 always has the lowest value in ANS. DC-Cloud has better performance both in ANC and in ANS.

To further analyze the simulation results, Fig. 8 and Fig. 9 give the total values of all five MCEs of the five service composition requests in TABLE VII. The X axis is the five composition methods. The Y axis in Fig. 8 is the average value of NC (ANC) of the five methods. The Y axis in Fig. 9 is the average value of NS (ANS) of the five methods. From Fig. 8, we find that Base-Clouds, Smart-Clouds and DC-Cloud have the lowest values in ANC, All-Clouds has the largest value in ANS. From Fig. 9, we find that COM2 has the lowest value in ANS, and followed by DC-Cloud. To COM2, DC-Cloud reduces 12 in NC under the wasting of about 1000 in NS totally. So, DC-Cloud gives a good performance in the NC, at the same time, it needs a shorter time to find a composition sequence, because it has a lower value in ANC.

VI. CONCLUSION AND FUTURE WORK

This paper gives a new service composition method under multi-cloud environment. We evaluate our method under multiple service composition requests and multiple Clouds. Simulations show our method not only reduces the average number of combined Clouds, but also ensures to get the service composition request in a short time.

REFERENCE

[1] Guobing Zou; Yanglan Gan; Jianxing Zheng; Bofeng Zhang, "Service composition and user modeling for personalized recommendation in Cloud computing," Computing, Communication and Networking Technologies (ICCCNT), 2014 International Conference on , vol., no., pp.1,7, 11-13 July 2014 doi: 10.1109/ICCCNT.2014.6963025

[2] Daniel A. Menascé, Honglei Ruan, Hassan Gomaa, QoS management in service-oriented architectures, Performance Evaluation, Volume 64, Issues 7–8, August 2007, Pages 646-663, ISSN 0166-5316, http://dx.doi.org/10.1016/j.peva.2006.10.001.

[3] Le Sun, Hai Dong, Farookh Khadeer Hussain, Omar Khadeer Hussain, Elizabeth Chang, Cloud service selection: State-of-the-art and future research directions, Journal of Network and Computer Applications, Volume 45, October 2014, Pages 134-150, ISSN 1084-8045, http://dx.doi.org/10.1016/j.jnca.2014.07.019.

[4] Breiter, G.; Naik, V.K., "A Framework for Controlling and Managing Hybrid Cloud Service Integration," Cloud Engineering (IC2E), 2013 IEEE International Conference on , vol., no., pp.217,224, 25-27 March 2013 doi: 10.1109/IC2E.2013.48

[5] Gutierrez-Garcia JO, Sim K. Agent-based Cloud service composition. Applied Intelligence, 2013,38(3):436-64.

[6] Guobing Zou; Yanglan Gan; Jianxing Zheng; Bofeng Zhang, "Service composition and user modeling for personalized recommendation in Cloud computing," Computing, Communication and Networking Technologies (ICCCNT), 2014 International Conference on , vol., no., pp.1,7, 11-13 July 2014 doi: 10.1109/ICCCNT.2014.6963025

[7] Jun Huang, Guoquan Liu, Qiang Duan, Yuhong Yan, QoS-Aware Service Composition for Converged Network-Cloud Service Provisioning, 2014 IEEE International Conference on Services Computing (SCC), DOI: 10.1109/SCC.2014.18 Publication Year: 2014, Page(s): 67 – 74

[8] Lianyong Qi, Ying Tang, Wanchun Dou, Jinjun Chen, Combining local optimization and enumeration for QoS-aware web service composition, in: IEEE International Conference on Web Services, 2010, pp. 34‑41.

[9] Daniel A. Menascé, Emiliano Casalicchio, Vinod Dubey, On optimal service selection in Service Oriented Architectures, Performance Evaluation, Volume 67, Issue 8, August 2010, Pages 659-675, ISSN 0166-5316, http://dx.doi.org/10.1016/j.peva.2009.07.001.

[10] Heba Kurdi, Abeer Al-Anazi, Carlene Campbell, Auhood Al Faries, A combinatorial optimization algorithm for multiple Cloud service composition, Computers & Electrical Engineering, Available online 13 December 2014, ISSN 0045-7906, http://dx.doi.org/10.1016/j.compeleceng.2014.11.002.