doi:10.3772/j.issn.1006-6748.2015.04.016

Service optimization in programmable cloud network①

Ding Hao (丁 浩), Yang Yang②, Mi Zhenqiang
(School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, P. R. China)

Abstract

With the rapid development in cloud data centers and cloud service customers, the demand for high quality cloud service has been grown rapidly. To face this reality, this paper focuses on service optimization issues in cloud computing environment. First, a service-oriented architecture is proposed and programmable network facilities are utilized in it to optimize specific cloud services. Then various cloud services are categorized into two subcategories: static services and dynamic services. Furthermore, the concepts of cloud service quality and cloud resource idle rate are defined, and the aforementioned concepts have also been taken into consideration as parameters in the service optimization algorithm to improve the cloud service quality and optimize system workload simultaneously. Numerical simulations are conducted to verify the effectiveness of the proposed algorithm in balancing the workload of all servers.

Key words: distributed cloud architecture, programmable network, cloud service quality, load balancing

0 Introduction

The booming Internet service in the last decade has become on essential part on people's daily life. Due to the limitations in service resources and the massive growth in customer's service requirements, the problem of how to effectively deliver high quality services with minimum costs has become an important issue.

A widely accepted solution to this problem is the utilization of cloud computing architecture. Cloud computing allows people to access data, application, content and other cloud services from a Web browser via the Internet hosted on the cloud platform[1]. Additionally, a cloud platform can normally integrate majority of service resources to deliver available services or composited services. Therefore, the cloud architecture has become a trend for the development of the whole information service industry.

To design a valid architecture of a cloud platform, the status of network traffic and customer distribution have to be considered. Therefore, with the growth in the number of cloud customers, recent years have witnessed growing research interests on distributed cloud architecture. Many efforts have been focused on the application of content delivery network (CDN) in cloud computing environment. A mathematical model of swarm-to-cloud applications in CDN environment was proposed in Ref. [2]. The applications namely peer-assisted content delivery services are provided. Another issue that has been extensively studied is content storage and replication, with a number of CDN based cloud storage schemes provided[3,4]. The aforementioned research has shown that distributed architecture is efficient in providing content-based cloud services.

The distributed cloud architecture has also brought new challenges and opportunities for the current network. Studies on future cloud network[5,6] gained a lot of concern due to the complex nature of numerous cloud services and the diversity of customers demand. Specifically, research on Programmable Network[7,8] (PN) and its application in cloud computing environment has become a hot spot. Programmable network is also called software-defined network, which means network facilities will be controlled by software from the application service layer instead of configuration interfaces from the network layer only. The feasibility of remote control from the service providers is the most significant feature of programmable network. Thus, service providers could change their control strategy based on the status of their own service. Programmable network models such as OpenFlow[9] have been widely studied and tested in both campus and data center net-

① Supported by the National Natural Science Foundation of China (No. 61272508, 61472033, 61202432).
② To whom correspondence should be addressed. E-mail: yyang@ustb.edu.cn
Received on Sep. 4, 2014

works.

It has been proved that, for certain services, the requirement of content-oriented cloud service can be fulfilled by utilizing the architecture of CDN and PN. Nevertheless, when dealing with complex and hybrid services (not only static content services or multimedia services) and crowded network status (like flash-crowd event[10]), none of the existing technologies are powerful enough to address such a problem.

In this work, the service optimization issues in cloud computing environment are focused on and a service-oriented architecture is proposed to address this issue. This architecture is fully distributed with an origin cloud data center (OCDC), which is the original and core server of the system, and several surrogate servers which are distributed at the edge of the network and known as edge cloud servers (ECS). Under this architecture, programmable network facilities are utilized to allow the real-time remote service control, which is dedicated to distinguish the service information and to control the network traffic load. Furthermore, the mathematical model of the distributed cloud architecture is presented and the definition and analysis of several important parameters are provided. Finally, to optimize the quality of cloud services as well as the entire system workload, a service optimization algorithm is also proposed. This algorithm takes the relevant service and workload parameters into consideration to fulfill the objective of optimization, and the complexity and convergence properties are also proved to verify the efficiency of the proposed algorithms. Numerical simulations have been also conducted to verify the effectiveness of the proposed algorithm.

The rest of this paper is organized as follows. Section 1 describes the system model of the proposed distributed cloud architecture. Section 2 provides the framework of the service optimization algorithm, while Section 3 introduces the service optimization algorithm. Section 4 is dedicated to numerical simulation and results. Section 5 summarizes this paper and shows the future work direction.

1 System model

To address the service optimization issue, a service-oriented architecture is proposed first. Fig. 1 shows the distributed cloud architecture with programmable network facilities. Under this architecture, the origin cloud data center holds the strongest capability of data processing and storage, while the edge cloud servers are topologically and physically close to cloud customers so they could significantly reduce network delay and jitter to guarantee the service quality of deployed cloud services. In addition, private communication channels are also utilized in the information transmission issues between the origin center and edge servers to ensure the transfer speed of service information and the reliability of service deployment.

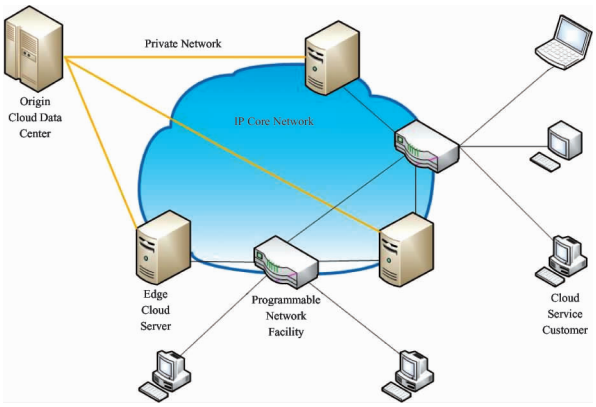

Fig. 1 Distributed programmable cloud architecture

Just like edge cloud servers, programmable network facilities are also deployed on the edge of the IP core network. So that they could get the information of QoS requirements from the end cloud customers directly and arrange the appropriate server to provide the services. Furthermore, all the programmable facilities in the same cloud could get and update the information of customer distribution, edge server workload and network traffic on run-time and adjust the service optimization algorithm synchronously without jeopardizing the traffic of the core network. The mathematic model of the proposed architecture is presented below:

Definition 1.1: Consider that asymmetry digraph $G = < V, E >$ represents the cloud architecture in Fig. 1. In the pair, V is the set of all network nodes, including cloud servers (V_S), programmable network facilities (V_P) and service customers (V_C), known as

$$V = V_S \cup V_P \cup V_C \tag{1}$$

E is the set of all network edges, including the private communication channels between origin cloud data center and edge servers (E_P), the network between edge server and programmable facilities (E_N) and the edge network between programmable facilities and cloud service customers (E_L):

$$E = E_P \cup E_N \cup E_L \tag{2}$$

All the edges work as communication links. The pairing relationship between V and E is subjected to Fig. 2.

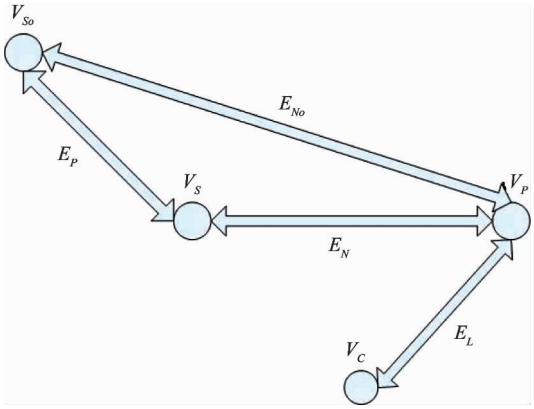

Fig. 2 Graphic model

In Fig. 2, V_{So} denotes origin cloud data center and E_{No} denotes the network between the origin cloud data center and programmable facilities. As could be seen from Fig. 2, there is no direct links between node V_C and V_S. All the information exchanged between server and customer should go through node V_P for V_P represents the function of programmable network facilities. The programmable facilities are dedicated to information collecting, service control and load balancing. Thus, they are the key nodes of the whole architecture. In terms of edges, although they are all Ethernet communication links, they differ a lot in the network bandwidth and data transmission rates. E_P represents the private channel among cloud servers, and it possesses an extremely high bandwidth and data transmission rate. Moreover, its upload and download service flows are almost the same. However, for E_N and E_L, they are asymmetry links with a huge download flow and a small upload flow. Additionally, with the frequent change in network service load, the data transmission speed of E_N also varies dynamically over time.

Under this architecture, it is an important issue for programmable facilities to choose an appropriate server for customers. The criteria for choosing proper service will depend on both the service requirement from the end-users and the resource condition from the cloud system. Relevant factors will be discussed in the following section.

2 Framework of service optimization algorithm

When a cloud customer requires services from the system, the request will be first sent to the programmable network facilities. Then the facilities will decide to choose an appreciate server for the customer or to refuse the request, and decide to choose a suitable server according to the following factors:

2.1 Service types and customer types

Programmable network facilities work on the service layer. Thus, they could recognize related service parameters and distinguish the service type of the required services.

All cloud services are defined into two service types:

Definition 2.1:

i) Static service is defined as services that could be provided by one edge cloud server independently.

ii) Dynamic service is defined as services that should be provided by several edge cloud servers or the origin data center collaboratively.

Static services involve content downloading services, online editing software and other services which could be replicated and deployed on any edge cloud servers without too much cost, while dynamic services are usually referred to real-time statistics services, big data services and large-scale Internet services, e. g., social network services. Compared with static services, dynamic services generally have a higher service quality requirement for the cloud servers due to the time synchronization and pattern diversity of the services.

Based on definition 2.1, cloud service customers are further classified into three types:

Definition 2.2:

i) C_a: a static service customer is defined as a customer with 75% or more in quantity of his required services are static services.

ii) C_d: a dynamic service customer is defined as a customer with 75% or more in quantity of his required services are dynamic services.

iii) C_h: a hybrid customer is defined as a customer with both static services or dynamic services requirements in quantity of 25% to 75%.

2.2 Service quality

When choosing an edge cloud server to deliver high quality services for the cloud customer, the physically or topologically closest one is used to be the most appropriate one. However, due to the complex characteristic of cloud services and network conditions, the service quality of the closest edge cloud server is not always the best especially in the flash-crowd moment. Thus, instead of physical location, the cloud service quality should be considered as the most important parameter when selecting an appropriate server.

As mentioned above, the quality of a cloud service is determined by both the service processing speed of the edge server and the data transmission speed in the edge network. So the cloud service quality could be

defined as:

Definition 2.3: The relative service quality Q_{S_i,C_j} of server S_i for a specific customer C_j could be calculated by

$$Q_{S_i,C_j} = \mu\omega_{S_i} + \upsilon\theta_{S_i,P_k} \tag{3}$$

where ω_{S_i} is the relative service processing speed of edge cloud server S_i, and θ_{S_i,P_k} is the relative data transmission speed in the edge network between server S_i and programmable facility P_k which connects directly with customer C_j. μ and υ were two weighted coefficients which represent the weights of the server and network separately.

In definition 2.3, higher value of Q_{S_i,C_j} means higher service quality. As an exceptional case of Eq. (3), the service quality of the origin cloud data center should be calculated by

$$Q_{S_0} = \mu\omega_{S_0} + \upsilon\theta_{N_0} \tag{4}$$

in which, θ_{N_0} is the data transmission speed of the core network.

2.3 Resource idle rate

As discussed in Section 1, cloud customers have a variety of service requirements and a considerable number of them make occupancy of the virtualized resources for very long time. Thus, it is essential for the system to keep workload of all the edge cloud servers and the edge network into a relatively balance. The resource idle rate is utilized to represent the workload status of one cloud server and its nearby edge network, and it is defined as:

Definition 2.4: The resource idle rate R_{S_i} of edge cloud server S_i could be calculated by

$$\begin{cases} R_{S_i} = \psi(1 - U_{S_i}) + \varphi(1 - U_{N,S_i,P_k}), \\ \qquad 0 < U_{S_i}, U_{N,S_i,P_k} < 1 \\ \psi + \varphi = 1, \psi > 0, \quad \varphi > 0 \end{cases} \tag{5}$$

when U_{S_i} is the utilization rate of edge cloud server S_i and U_{N,S_i,P_k} is the utilization rate of the edge network between server S_i and programmable facility P_k which connects directly with customer C_j.

As an exceptional case of Eq. (5), the relative resource idle rate of the origin cloud data center should be calculated by

$$\begin{cases} R_{S_0} = \psi U_{S_0} + \varphi U_{N_0}, \quad 0 < U_{S_0}, U_{N_0} < 1 \\ \psi + \varphi = 1, \quad \psi > 0, \varphi > 0 \end{cases} \tag{6}$$

in which, U_{N_0} is the relative utilization rate of the core network.

Note that these three parameters contain critical information of the customer, the server and the network. They are necessary for the optimization of both the customer service quality and system workload.

3 Service optimization algorithm

3.1 Main idea and steps

The service optimization algorithm utilizes above parameters to choose an appropriate server for the cloud service customer when the customer requires for deployed cloud services. Specifically, the rules below should be carefully considered when designing such an algorithm:

1) From the customer side, the algorithm should take the specific requirement from the cloud customer and find the most appropriate server with the best relative service quality. However, from the side of cloud service provider, the "best" server for customer might not be the most appropriate for the system. So, the service optimization algorithm should consider both the service quality and workload balancing of servers and networks at the same time.

2) Due to the different characteristics of service type and customer type, edge servers should serve mostly for static services. On the contrary, the origin data center should mostly serve for dynamic service customers. However, service quality and workload condition should still be considered as the primary factors.

3) When choosing an appropriate server, those who are busy at that moment or their relative service quality are too low to meet the requirement of the customer should be filtered out and the resource of the origin data center should be massive enough to support the edge servers and keep the whole system into a stable condition.

Based on the analysis above, two new parameters Y and η are made use of to represent the appropriate degree of a cloud server. Actually, Y is the product of service quality Q and resource idle rate R. η is a percentage.

Taking all these into consideration, Table 1 shows the pseudo code of the service optimization algorithm. The main steps of the algorithm are also given in Fig. 3.

The proposed algorithm aims at improving the cloud service quality and at the same time optimizing the server and network workload. Instead of relying on any absolute value, relative service quality between servers and customers is simply utilized to improve the efficiency of the algorithm. Moreover, as comparison is conducted on the service quality and resource idle rate between the origin data center and the selected edge servers, the proposed algorithm could guarantee that

dynamic services would be delivered by the origin cloud data center as many as possible. Meanwhile, static services should be served mostly by edge servers.

The algorithm is further explained by examples below.

Table 1 Service optimization algorithm

Service optimization algorithm for cloud customer

Server selection and optmization (Q_{Cj}, C_T, $Q_{Si,Cj}$, Q_{So}, R_{Si}, $R_{So}Y_i$, η)

check all severs, compute R_{So} and R_{Si} of all edge servers, find if S_i is busy;
compute Q_{So} and $Q_{Si,Cj}$ of all edge servers;
compare $Q_{Si,Cj}$ with Q_{Cj}, find out available servers l of $S_l \in \{S_l \mid Q_{Sl,Cj} > Q_{Cj}$ && S_l is not busy$\}$;
if (S_l = NULL)
    service rejects, algorithm ends;
else if ($S_l = \{S_o\}$)
    identify the type of customer j;
    if ($C_T = C_a$ or $C_T = C_h$)
        service rejects, **algorithm ends**;
    else if ($C_T = C_d$)
        selectedServer = S_o, **algorithm ends**;
else **algorithm continue**;
identify the type of customer j;
if ($C_T = C_a$)
    compute Y_l of $\{Y_l \mid Y_l = Q_{Sl,Cj} \times R_{Sl}\}$;
    compare and find server m of $Y_m = \max\{$ all values of $Y_l\}$;
    selectedServer = S_m, **algorithm ends**;
else if ($C_T = C_d$)
    compute Y_l of $\{Y_l \mid Y_l = Q_{Sl,Cj} \times R_{Sl}\}$, Y_o of $\{Y_o = Q_{So} \times R_{So}\}$;
    compare and find server m of $Y_m = \max\{ Y_o$ and all values of $Y_l\}$;
    find out any server g of $S_g \in \{S_g \mid \eta Y_m < Y_g \leqslant Y_m\}$;
    if ($S_o \in \{$ any server $S_g\}$)
        selectedMainServer = S_o;
        backupServers = S_h, $S_h \in \{S_h \mid \eta P_m < P_h \leqslant P_m$ && $S_h \neq S_o\}$.
    else if ($S_o \notin \{$ any server $S_g\}$)
        compare and find server w of $Q_{Sw,Cj} = \max\{ Q_{Sg,Cj}$ of any server $S_g\}$;
        selectedMainServer = S_w;
        backupServers = S_h, $S_h \in \{S_h \mid \eta P_m < P_h \leqslant P_m$ && $S_h \neq S_w\}$.
    end
else if ($C_T = C_h$)
    compute Y_l of $\{Y_l \mid Y_l = Q_{Sl,Cj} \times R_{Sl}\}$, Y_o of $\{Y_o = Q_{So} \times R_{So}\}$;
    compare and find server m of $Y_m = \max\{Y_o$ and all values of $Y_l\}$;
    find out any server g of $S_g \in \{S_g \mid \eta Y_m < Y_g \leqslant Y_m\}$;
    if ($S_o \in \{$ any server $S_g\}$
        and $Q_{So} = \max\{Q_{Sg,Cj}$ of any server $S_g\}$
        and $R_{So} = \max\{R_{Sg}$ of any server $S_g\}$)
        selectedMainServer = S_o;
        backupServers = S_h, $S_h \in \{S_h \mid \eta P_m < P_h \leqslant P_m$ && $S_h \neq S_o\}$.
    else selectedMainServer = S_m;
        backupServers = S_h, $S_h \in \{S_h \mid \eta P_m < P_h \leqslant P_m$ && $S_h \neq S_m\}$.
    end
end

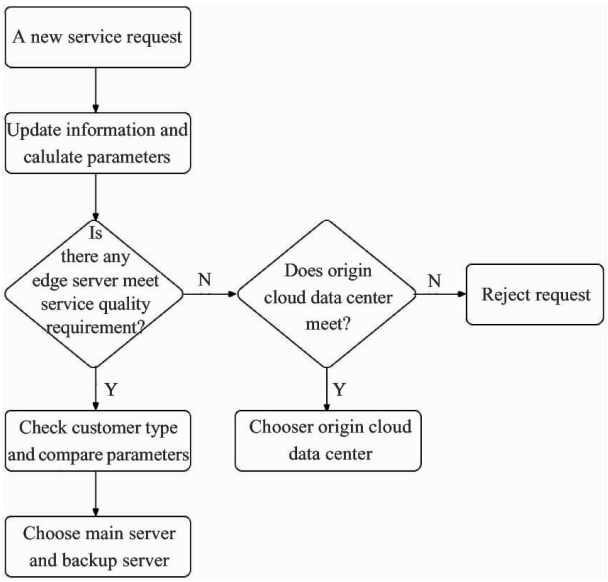

Fig. 3 Algorithm flowchart

3.2 Case study

In these two cases, it is assumed that the servers shown in the corresponding figure are the only servers that meet the service quality Q of the customer.

Assume that service customer j_1 sends a request for an online editing service to programmable facility P_1. Obviously it is a static service customer. So P_1 updates information and computes resource idle rate R and service quality Q according to the latest data, and finds out available servers (line1, 2). For j_1, if available servers are S_0, S_1 and S_2, as shown in Fig. 4, then P_1 will transfer to line 12, and identify the customer type (line12, 13). j_1 is a static customer, so the appropriate server will be selected in line 14 to 17. As for static service customers, the appropriate server should be the one with the highest Y value among edge servers, not including the origin data center. So the selected server for j_1 is S_2 for Y_2 value equals 24 in Fig. 4.

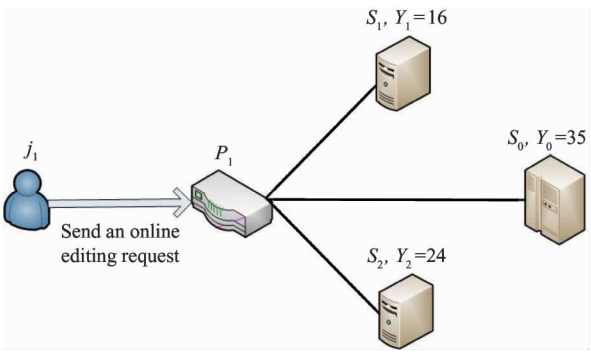

Fig. 4 Static service customer

Assume that there is a customer known as j_2 that has sent a social network service request to programmable facility P_2, and the status of available servers with their Y value as well as value η are as illustrated in Fig. 5. For dynamic service customer, the process of the algorithm works the same compared with static service customer in line 1 to 13. As defined, customer j_2 is a dynamic service customer, and the selection algorithm in line 18 to 28 will make the optimization process different from static service customers. As we can see from Fig. 5, the best server with a highest Y value is S_3. But Y_0 is also among the best servers ($\eta \times Y_3 < Y_0 < Y_3$). So according to line 22 and 23, the selected main server is S_0. As discussed in Section 1, for some dynamic services, they should be provided by several edge cloud servers or the origin data center collaboratively. So the algorithm further chooses backup servers and in this case it is S_3. S_4 is not a backup server for $Y_4 < \eta \times Y_3$.

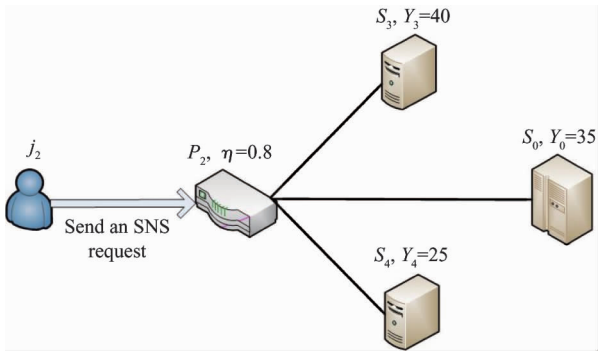

Fig. 5 Dynamic service customer

For hybrid customers, the algorithm holds a high degree of consistency with dynamic customers. The only difference is that for a hybrid service customer, S_0 would be chosen as the appropriate server only when both Q_0 and R_0 are the max value of all.

3.3 Performance analysis

In this part, the performance of the proposed algorithm is analyzed, which includes average time complexity and convergence.

(1) Average time complexity

The average time complexity of the proposed algorithm is $O(nlogn)$, and n represents the number of cloud servers.

Prove: The workload of the algorithm mainly comes from two sources:

(i) The data updating and calculation from a great number of servers. In Table 1, line 1, 2, 15, 19 and 30 contribute to this. The average time complexity of line 1 and 2 is a polynomial complexity $O(n)$, and the average time complexity of line 15, 19 and 30 is less than $O(n)$ because the computed server number is

less than n.

(ii) Comparisons between calculated results. In Table 1, line 3, 16, 20, 26 and 31 contribute to this. The classical heapsort algorithm is ideally suitable in the comparison. As proved in Ref. [11], the average time complexity of heapsort algorithm is $O(nlogn)$.

Thus, the time complexity of the proposed algorithm is determined by the sorting part of the algorithm and the average time complexity of the proposed algorithm is $O(nlogn)$.

(2) Algorithm convergence

The proposed algorithm must be convergent. That is to say, the programmable facilities would refuse the service request or ultimately find an appropriate server for any types of cloud customer.

Prove: The reject condition has been considered in line 4 to 11 of the proposed algorithm. Additionally, line 14, 18 and 29 have enumerated all the cases of customer types. Thus, the proposed algorithm is convergent.

4 Numerical simulation and results

The numerical simulation of this paper is established and studied in MATLAB. In the study, it is assumed that the initial utilization rate of the entire network is the same between any server i and any customer j, to test the performance of the service optimization algorithm in balancing the server workload of the distributed architecture. Detailed parameter settings, results and analysis are presented below:

4.1 Parameter settings

The simulation defines 300 stochastic customers which are subjected to Poisson distribution. Among them, each type of customers accounts for one third. Additionally, 5 edge cloud servers and the origin cloud data center are selected. Detailed parameter settings are shown in Table 2.

Table 2 Simulation parameter settings

Parameter	Description	Value/Range
C_T	The type of customers	C_a, C_d, C_h
S_i	Edge cloud servers	S_1, S_2, S_3, S_4, S_5
S_o	The origin cloud data center	S_o
G	The maximum capacity of S_o and S_i	$\{S_o:1500\}$; $\{S_i: 300\}$
Q_{Cj}	Service quality requirement of customer j	40 ~ 80
$Q_{Si,Cj}$	The relative quality of services which are offered by edge cloud server i for customer j	30 ~ 100
Q_{So}	The relative service quality of the origin cloud data center	85
U_{Si}	The initial utilization rate of edge cloud server S_i	$\{S_1:30\%\}$; $\{S_2:40\%\}$; $\{S_3:50\%\}$; $\{S_4:60\%\}$; $\{S_5:70\%\}$
$U_{N,Si,Cj}$	The initial utilization rate of the edge network between server i and customer j	50%
U_{So}	The initial utilization rate of the origin cloud data center S_o	50%
U_{No}	The initial relative utilization rate of the core network	50%
$\mu:\nu$ / $\psi:\varphi$	The ratio of the weight coefficients between servers and networks	0.75:0.25
η	The percentage parameter utilized in the algorithm	0.8

Note that as relatively consistent value is used to make sure the weight coefficients work properly.

4.2 Results and analysis

Due to the randomness of customer distribution and the unpredictability of the value of relative service

quality before the experiment, the statistic of the status of service quality is done first. For a specific customer, server i is called a "best-service-quality server" when the quality of offered services from server i is the best. Table 3 shows the amount of the three types of customers whose best-service-quality server is server i.

Table 3 Customer amount with highest QoS in each server

Server number	S_0	S_1	S_2	S_3	S_4	S_5
Amount of static service customers	*N/A*	18	21	18	24	19
Amount of dynamic service customers	82	3	6	2	4	3
Amount of hybrid customers	78	6	6	2	3	5
In total	160	27	33	22	31	27

Fig. 6 shows the final amount of each type of customers allocated on each server in a bar chart. As can be seen, most of the dynamic service customers and a large number of hybrid customers are allocated to the origin cloud data center S_0. Moreover, servers with a relatively low utilization rate (like S_1 or S_2) are allocated with several times of customers compared with servers with a relatively high utilization rate (like S_5).

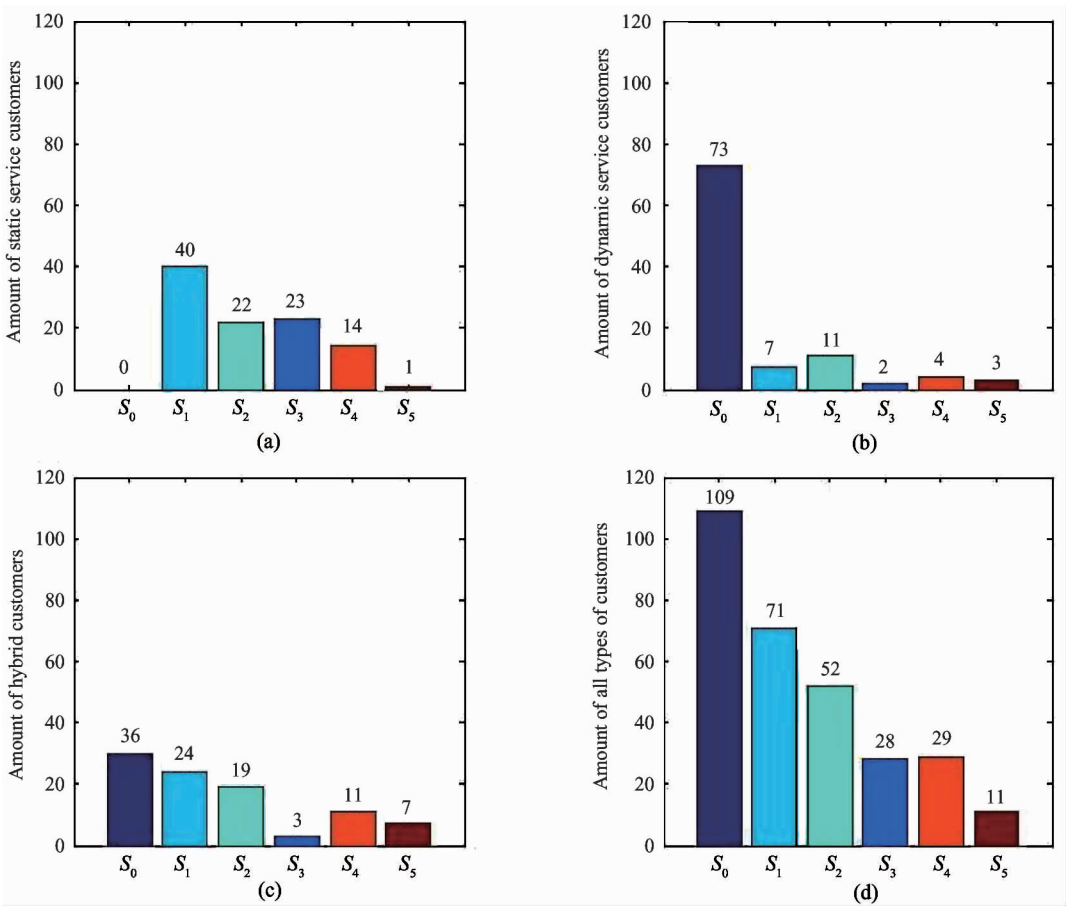

Fig. 6 The amount of customers allocated on each server

By statistics and calculations for the allocation results, Table 4 shows the comparison of the server utilization rate in initial and ultimate time. The idlest server S_1 in initial time witnesses the greatest increment (23.67%), while the busiest server S_5 in initial time only sees a very slight increment (3.67%). Thus, the proposed algorithm effectively controls the workload increasing pace of high-load servers, making it tend to be balanced. Moreover, while according to Table 3 that compared to S_3, S_4 is the highest QoS server for more customers, its workload increment in Table 4 does not significantly exceed S_3, which proves that the algorithm controls the workload increment of S_4.

Table 4 Comparison of server utilization rate

Server number	S_0	S_1	S_2	S_3	S_4	S_5
Initial utilization rate	50%	30%	40%	50%	60%	70%
Ultimate utilization rate	57.27%	53.67%	57.33%	59.33%	69.67%	73.67%

5 Conclusion and future work

In this work, a distributed architecture for cloud service customers is proposed, along with a service optimization algorithm. This architecture deploys the edge cloud servers on the edge of the network with programmable network facilities. The programmable network facilities could identify the customer type, which is an important parameter used in the proposed service optimization algorithm. The service optimization algorithm also takes another two important parameters: service quality and resource idle rate. Simulation results show that the proposed algorithm could optimize the server selection process and effectively balance the workload among all servers.

According to the discussion in Section 1 and the simulation results, it is noticed that the resource idle rate could affect the service quality of an edge server, and the relative service quality will affect the number of new coming customers. Mathematical relationship among them might be a direction for the research work in the future.

References

[1] Tran H A, Mellouk A, Hoceini S. QoE content distribution network for cloud architecture. In: Proceedings of the 2011 First International Symposium on Network Cloud Computing and Applications (NCCA), Toulouse, France, 2011. 14-19

[2] Albanese F, Carra D, Michiardi P, et al. Cloud-based Content Distribution on a Budget, BUCS-TR-2010-022, Boston University: Computer Science Department, 2010

[3] Broberg J, Buyya R, Tari Z. MetaCDN: Harnessing 'Storage Clouds' for high performance content delivery. *Journal of Network and Computer Applications*, 2009, 32(5): 1012-1022

[4] Wang Y, Wen X, Sun Y, et al. The content delivery network system based on cloud storage. In: Proceedings of the 2011 International Conference on Network Computing and Information Security (NCIS), Guilin, China, 2011, 1: 98-102

[5] Shang Y, Li D, Xu M. Energy-aware routing in data center network. In: Proceedings of the 1st ACM SIGCOMM Workshop on Green Networking, New Delhi, India, 2010. 1-8

[6] Benson T, Akella A, Shaikh A, et al. CloudNaaS: a cloud networking platform for enterprise applications. In: Proceedings of the 2nd ACM Symposium on Cloud Computing, Cascais, Portugal, 2011: 8

[7] Yuan L, Chuah C N, Mohapatra P. ProgME: towards programmable network measurement. *IEEE/ACM Transactions on Networking* (*TON*), 2011, 19(1): 115-128

[8] Chowdhury N M, Boutaba R. A survey of network virtualization. *Computer Networks*, 2010, 54(5): 862-876

[9] Sherwood R, Chan M, Covington A, et al. Carving research slices out of your production networks with OpenFlow. *ACM SIGCOMM Computer Communication Review*, 2010, 40(1): 129-130

[10] Jung J, Krishnamurthy B, Rabinovich M. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In: Proceedings of the 11th International Conference on World Wide Web, Honolulu, USA, 2002. 293-304

[11] Williams J W J. ALGORITHM-232-HEAPSORT. *Communications of the ACM*, 1964, 7(6): 347-348

Ding Hao, born in 1987. He started pursing his Ph. D degree in School of Computer and Communication Engineering, University of Science and Technology Beijing (USTB) from 2010. He also received his B. S. degree from USTB in 2010. His research interests include network optimization, service optimization and network measurement.