Towards Network-Aware Service Composition in the Cloud

Shangguang Wang, *Senior Member, IEEE*, Ao Zhou, Fangchun Yang, *Senior Member, IEEE*, Rong N. Chang, *Senior Member, IEEE*

Abstract——Composing several API-defined services into one composite service per user requirements has become an important service creation approach in the cloud-enabled API economy. Various service selection approaches in support of service composition on demand have been proposed. They usually assume that networking resources are over-provisioned and their usage needs not be considered when making quality-aware service composition decisions. In practice, these approaches often lead to wasteful network resource consumption and impractical end-to-end QoS optimality for cloud-based services. This paper proposes a network-aware cloud service composition approach, named NetMIP, with comparative experimental evaluations for the clouds that adopt the widely deployed fat-tree network topology. By formalizing the service composition goal as a multi-objective constraint optimization problem, we have validated the proposed approach can be used to effectively reduce network resource consumption and deliver QoS optimality while satisfying the end-to-end QoS constraints for the candidate composite services in the cloud. The comparative experimental evaluations are done via a credible cloud infrastructure simulation system, named WebCloudSim. Extensive evaluation results show that NetMIP outperforms several representative cloud service composition approaches in terms of network resource consumption, QoS optimality, and computation time under various service selection workloads and fat-tree network topology settings.

Index Terms—cloud computing; service composition; fat-tree cloud data center network; network resource consumption; QoS

1 Introduction

Rapid innovation of infrastructure-as-a-service clouds has enabled the delivery of many consumer and enterprise application capabilities through the Internet as API-defined services with no need to expose service implementation details. In-house and commercial *cloud service provider* (CSP) marketplaces (hosted by Alibaba, Amazon, IBM, etc.) have formed a solid foundation for a cloud-based API economy in which computing and data resources can be managed and exploited cost-efficiently by dynamically composing the service APIs in production [1]. For instance, an on-line shoes shopping service can be composed via a preference-learning service, a search service, a payment service, and a shipping service.

Service composition is a well-established research field [2] and many relevant techniques have been proposed in support of various service composition constraints [3]. In terms of the *service-oriented architecture* (SOA) triangle [4], *service providers* publish their services in a *service registry* and a *service discovery* component returns a set of *concrete services* that have the requested functional (e.g., search) and non-functional (e.g., QoS) properties. After selecting a specific qualified concrete service, the *service requestor* can consume the service's capabilities per its API invocation rules. A service requestor can be a *service broker* that serves its users or clients via several classes of concrete services. It is important for a service broker (e.g., a CSP) to be able to make optimal service composition decisions and do that in a scalable manner when the demand for provisioning composite services is high and the number of concrete services is large.

Most existing service composition schemes for cloud services [5-10] merely reuse traditional web service composition technology [11-15], which considers maximum aggregated QoS values (e.g., throughput and delay) of concrete services with impractical end-to-end QoS optimality. Moreover, they are ignorant of the excessive network resource consumption issues that could be caused by poor decisions on where the chosen concrete services run on the cloud datacenter networks. These two motivating problems of this paper are elaborated below.

1) Consider Network Resource Consumption. Most of today's cloud datacenter networks adopt a multi-rooted tree topology structure called the *fat-tree* approach [16], which delivers large bisection bandwidth through rich path multiplicity [17] as shown in Fig. 1. In the fat-tree topology structure, all servers that are physically connected to the same *edge switch* form their own *subnet*. All physical servers that share the same *aggregation switches* are in the same *pod*. One physical server can host several VMs, and one VM can host several services. The top layer is the core tier, and the switches in this layer are *core switches*. A link connecting a core switch and an aggregation switch is a *core link*. Because all cross-datacenter traffic must be routed through the core switches, utilization of the core

- *S. Wang, A. Zhou, and F. Yang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. E-mail: {sgwang; aozhou; fcyang}@bupt.edu.cn.*
- *R. N. Chang is with IBM Research. E-mail: rong@us.ibm.com.*

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2603504, IEEE Transactions on Cloud Computing

links must be properly managed from the viewpoint of cloud service quality assurance (e.g., unnecessary consumption of the core links should be minimized whenever possible). Then, as due to increases in data-intensive services/applications in the cloud data center, cloud data center traffic is experiencing rapid growth[1]. Therefore, in cloud data centers shared by many data-intensive services, the upper-level network resource (i.e., bandwidth resources), especially the network resources of core layer, of the cloud data center network may become a bottleneck [18]. Moreover, we note that most existing cloud service composition approaches select and compose services based upon recorded QoS properties of candidate services without taking into account the consumption of necessary physical servers, subnets, and/or pods. They would not only result in excessive network resource consumption (particularly for the core links), but also impede on-demand scaling for provisioning quality-assured composite-services.

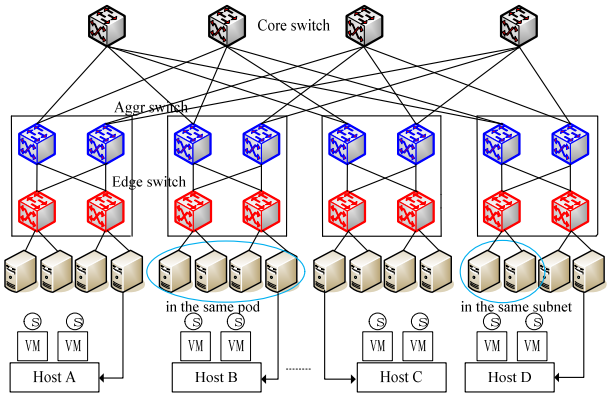


Fig. 1. Fat-tree topology structure. The switches at the top (black), middle (blue), and bottom (red) layers are core, aggregation, and edge switches, respectively. S presents services running on one VM. In this paper, all services are from the same CPS in the same cloud data center.

2) No End-to-End QoS Optimality. In the conventional service composition scheme, the end-to-end QoS optimality (hereafter referred to as *QoS optimality*) of a service composition depends on the aggregated QoS of the individual services [19]. The QoS of an individual service can be obtained by calculating the QoS history of a service that is invoked by other services or consumers. However, the QoS aggregation rules focus only on the QoS of individual services and do not consider the QoS of the networking infrastructure (e.g., *network latency*, *network throughput*, etc.) between multiple concrete services during service composition. With reference to Fig. 1, there could be one composite service composed of two VM-based concrete services running on hosts A and D, respectively. As per conventional service composition schemes, the service selection decision would be optimal in terms of the QoS constraints used for selecting them from the qualified candidate services. However, QoS experience with the composite service could gravely fall short of the promise due to, e.g., congested core links between the two services. In practice, unrealistic QoS optimality would cause unexpected service composition failures and result in service-level agreement (SLA) [20] violations and revenue loss for CSPs.

Aiming at resolving the aforementioned problems for CSPs, we propose a network-aware service composition approach that optimizes service composition decisions by taking into account, among other decision-making factors, network resource consumption in fat-tree cloud datacenter networks and realistic (end-to-end) QoS optimality. To the best of our knowledge, the proposed approach is the first research effort of its kind with the novel contributions listed below:

1) In contrast to past research efforts in service composition, our work integrates composite service model and datacenter network topology, considers impact on network resource consumption according to service deployment locations, and assures realistic QoS optimality by factoring in network QoS properties in the proposed service composition process.
2) We present a network-aware service composition approach that contains a function for designing network resource consumption, computing the QoS of networks, proposing a network-aware service composition model, and finding the optimal solution with minimal network resource consumption and realistic QoS optimality.
3) We have quantitatively validated our approach is superior to several representative cloud service composition approaches (via the cloud simulation system WebCloudSim) in terms of network resource consumption, QoS optimality, and computation time under various service selection workloads and fat-tree network topology settings.

The remainder of this paper is organized as follows. Section 2 introduces related work. Section 3 presents our network-aware service composition approach, Section 4 illustrates comparative experimental evaluation results. We conclude the paper in Section 5 with an outlook on our future work.

2. Related Work

A number of schemes have been proposed for service composition in the cloud [3], which consists of horizontal service composition (dealing mainly with web/software services composed together according to a workflow or business process [8]) and vertical service composition

[1] *http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf*

(which takes the whole hardware, middleware, and software stack of a service into account). We note that both types of service composition derive from conventional QoS-aware web service composition [14,15].

Regarding horizontal service composition in the cloud, Gutierrez-Garcia and Sim [21,22] proposed an agent-based approach to composing different types of cloud services for multi-cloud environments. Each cloud participant (e.g., consumers, brokers, and service providers) and web service is represented by an agent, and agent-based cooperative problem-solving technique is used to dynamically select the most appropriate services (e.g., the cheapest services) to create a composite service. Like conventional QoS-aware web service composition approaches (e.g., [23-28], [29]), this approach [21,22] does not consider the impact of overloaded cloud infrastructure resources (e.g., VMs, physical servers, data center networks, etc.). In contrast, Dastjerdi and Buyya [9] considered virtual appliances (software images) and virtual machines as cloud services, and achieved the best combination of compatible virtual appliances and virtual machines that minimizes deployment cost and deployment time while maximizing reliability and adhering to compatibility constraints. Although this approach employs a multi-objective algorithm and fuzzy logic to enable users to conveniently express their (concrete) service selection preferences, it involves VM settings that are idealistic and impractical for cloud infrastructure resources (e.g., it does not associate the services with physical servers or fat-tree datacenter networks). Similarly, Kritikos and Plexousakis [10] presented a cloud service composition approach for multi-cloud applications such that an optimal service composition solution could be found to satisfy all end-user requirements with all possible design choices considered. Although this approach considers VM characteristics and cost in a more elaborate manner (e.g., number of cores, main memory size, and storage size) than [9], this work and similar work [30-32] are still considered impractical due to their ignorance of VMs and service locations in cloud datacenter networks [33].

Regarding vertical service composition in the cloud, Mietzner et al. [3] proposed a framework of vertical component composition based upon an enterprise service bus infrastructure to facilitate reusing the service components of a distributed application at layers of Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). Based on the framework, Karim [8] proposed a model for predicting end-to-end QoS values of cloud-based software solutions composed via SaaS and IaaS services. Although it exploits VM configuration settings, it does not consider service location and cloud datacenter network topology, which are key to the QoS assurance of composite cloud services. Jun et al. [34] proposed a model for converged network-cloud service provisioning, formulated the problem of QoS-aware network-cloud service composition as a type of multi-constraint optimal path, and developed an heuristics to solve the formulated problem. Compared with aforementioned approaches, this work aimed at delivering optimal end-to-end QoS across both networking and computing domains and attempted to obtain realistic QoS optimality for the cloud-hosted composite services. However, this work and other work [35] still could not achieve realistic QoS optimality owing to their insufficient exploitation of the deployment dependence between VMs and datacenter networks.

Compared with existing cloud service composition approaches, which exhibit excessive network resource consumption and impractical QoS optimality, our approach, by exploiting service locations in fat-tree cloud datacenter networks, strives to make the best service composition decisions while also satisfying minimal network resource consumption and realistic QoS optimality in the cloud.

3. Network-aware Service Composition

To facilitate presenting the proposed approach, we first define, in Section 3.1, some other terms and notations that will be used for the rest of the paper. We describe the network source consumption function used by our service composition algorithm in Section 3.2. In Section 3.3, we establish a QoS utility function, including a QoS utility function for services and a QoS utility function for networks. Via the network resource consumption function and the QoS utility function, we formalize the problem of network-aware cloud service composition and formulate the optimal solution as a minimum in a multi-objective optimization problem in Section 3.4. Finally, in Section 3.5, we describe the 0-1 mixed integer program we use for finding the optimal cloud service composition solution with minimal network resource consumption and realistic QoS optimality.

3.1 Preliminaries

In this paper, the target cloud environment exploits a fat-tree cloud datacenter, as shown in Fig. 1. It consists of three-level trees of switches and a set of physical servers that have different resource capacities.

The top layer is composed of c core switches which provide connections between aggregation switches; c denotes the number of core switches. The middle layer is composed of a aggregation switches which provide connections between edge switches; a denotes the number of aggregation switches. The bottom layer is composed of e edge switches which provide connections for physical servers; e denotes the number of edge switches. Without losing the generality of our approach, we assume every edge switch can be connected by p physical servers; i.e.,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2603504, IEEE Transactions on Cloud Computing

the datacenter network can provide connections for up to $N = pe$ physical servers.

TABLE I. NOTATIONS

Symbol	Meaning
c	Number of core witches
a	Number of aggregation switches
e	Number of edge switches
p	Number of physical servers that link to one edge switch
N	Total number of all physical servers in a fat-tree data center network
ps_i	A set of physical servers that links to the i^{th} edge switch
vm_i	A set of VMs in the i^{th} physical server
V	Number of VMs that run on a physical server
$\overline{s_i}$	A set of services in the i^{th} VM
S	The service composition in the cloud
s_i	The i^{th} service class
M	Number of service classes
N	Number of candidate services in a service class
$Packet_{core}$	Total size of packets transferred by core switches
$Packet_{agg}$	Total size of packets transferred by edge switches
$Packet_{edge}$	Total size of packets transferred by edge switches
$NRS(S)$	Total network resource consumption of service composition in the cloud
$q_k(s)$	The k^{th} attribute value in service s
$q_k(S)$	Aggregated service QoS values by the k^{th} attribute values from service composition in the cloud
r	Number of services/network QoS attributes
$U(S)$	QoS utility function of a service according to all attributes from service composition in the cloud
$UN(S)$	QoS utility function of a network according to all attributes from service composition in the cloud
C	A given set of global QoS constraints

1) *Physical server*. Let $ps_i = \{ps_{i1}, ps_{i2}, \dots, ps_{ip}\}$ denote a set of physical servers that links to the i^{th} edge switch. Each physical server can run multiple VMs, and is characterized by the CPU performance (specified by the number of MIPS), the amount of memory used, network bandwidth, and disk storage.
2) *Virtual Machine.* Let $vm_i = \{vm_{i1}, vms_{i2}, ..., vm_{iv}\}$ denote a set of VMs in the i^{th} physical server, where v denotes the number of VMs in the physical server. We assume each physical server hosts v VMs. One VM can run multiple services, and each service runs on only one VM as a time-varying workload.
3) *Service*. Let $\overline{s_i} = \{\overline{s}_{i1}, \overline{s}_{i2}, ..., \overline{s}_{il}\}$ denote a set of services in the i^{th} VM, where l denotes the number of services in the VM. We assume each VM runs l services while satisfying its resource allocation constraints (e.g., CPU, memory, and bandwidth).
4) *Service composition, service class, candidate service, and concrete service*. Let $S = \{s_1, s_2, \dots, s_m\}$ denote a service composition in the cloud that satisfies a specific set of user requirements. It comprises of a set of services, each of which is selected from a *service class*. A service class $s_i = \{s_{i1}, s_{i2}, ..., s_{in}\} (s_i \in S, 1 \le i \le n)$ contains a set of candidate services, where n is the number of candidate services. All of the candidate services in a service class provide the same capabilities, though different candidate services may have different QoS attributes. Every service composition solution can choose only one candidate service from a qualified service class and make that a concrete service for the solution.

TABLE I lists all of the aforementioned notations. In this paper, we consider the sequential composition model only since other models (e.g., parallel, conditional, and loop models) can be transformed into the sequential model using the techniques in [36,37].

3.2 Network Resource Consumption

Definition 1 **(Network resource consumption)**. *Network resource consumption* (NRS) is the total size of packets transferred by all switches in a fat-tree cloud data center network over a period time, which can be calculated as follows [38]:

$$NRS = Packet_{core} + Packet_{agg} + Packet_{edge} \quad (1)$$

where $Packet_{core}$ denotes the total size of packets transferred by core switches, which can be calculated as follows:

$$Packet_{core} = \sum_i x_{i,j} \times size(packet_i) \quad (2)$$

where $x_{i,j} \in \{0,1\}$, and $x_{i,j} = 1$ indicates that the j^{th} link is selected to transfer the i^{th} packets via the core switches; otherwise $x_{i,j} = 0$.

$Packet_{agg}$ denotes the total size of packets transferred by the aggregation switches, which can be calculated as follows:

$$Packet_{agg} = \sum_i y_{i,j} \times size(packet_i) \quad (3)$$

where $y_{i,j} \in \{0,1\}$, and $y_{i,j} = 1$ indicates that the j^{th} link is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2603504, IEEE Transactions on Cloud Computing

selected to transfer the i^{th} packets via the aggregation switches; otherwise $y_{i,j}=0$.

$Packet_{edge}$ denotes the total size of packets transferred by the edge switches, which can be calculated as follows:

$$Packet_{edge}=\sum_{i} z_{i,j}\times size(packet_i) \quad (4)$$

where $z_{i,j}\in\{0,1\}$, and $z_{i,j}=1$ indicates that the j^{th} link is selected to transfer the i^{th} packets via the aggregation switches; otherwise $z_{i,j}=0$.

In order to calculate network resource consumption for service composition in the cloud, we use $NRS(S)$ to denote the network resource consumption of service composition S as follows:

$$NRS(S)=\sum_{s_i\in S}(Packet_{core}+Packet_{agg}+Packet_{edge}) \quad (6)$$

where we denote the total size of packets transferred by all switches for the service candidate of service class $s_i(1\le i\le n)$ as the network resource consumption of service composition S in the cloud.

3.3 QoS Utility Function

3.3.1 QoS utility function of service

Consider a QoS requirement for service s with *r* attributes and attribute vector $qs=\{q_1(s),q_2(s),...,q_r(s)\}$, where the value of $q_k(s)$ $(1\le k\le r)$ represents the k^{th} attribute value in service s. Similarly, $q_k(S)$ is aggregated by the k^{th} attribute values from service composition S according to the QoS aggregation functions (more detailed description of the functions can be found in [7,14,15]).

In order to shield different units or scopes of QoS attributes, we adopt a QoS utility function [15] to map the vector of QoS values qs into the domain [0, 1] for uniform computation of multi-dimensional QoS attributes in a VM dependent manner, as shown in Definition 2.

Definition 2 (QoS utility function of service): The QoS utility functions for one service $s\in s_i(1\le i\le m)$ and service composition S are defined as follows [7,13-15]:

$$U(s)=\sum_{k=1}^{r}\frac{Q_{j,k}^{max}-q_k(s)}{Q_{j,k}^{max}-Q_{j,k}^{min}}.\omega_k, \quad (7)$$

$$U(S)=\sum_{k=1}^{r}\frac{Q_{k}^{max}-q_k(S)}{Q_{k}^{max}-Q_{k}^{min}}.\omega_k, \quad (8)$$

with

$$\begin{cases} Q_k^{max}=\sum_{j=1}^{m}Q_{j,k}^{max},\ Q_{j,k}^{max}=\max_{\forall s_{ji}\in s_j\in\overline{s}} q_k(s_{ji}) \\ Q_k^{\min}=\sum_{j=1}^{m}Q_{j,k}^{\min},\ Q_{j,k}^{\min}=\min_{\forall s_{ji}\in s_j\in\overline{s}} q_k(s_{ji}) \end{cases}, \quad (9)$$

where $w_k\in R^+(\sum_{k=1}^{r}w_k=1)$ represents the weight of each QoS attribute, $Q_{j,k}^{max}$ is the maximum value of the k^{th} attribute in all candidate services in the i^{th} service class s_i, $Q_{j,k}^{min}$ is its minimum value, Q_k^{max} and Q_k^{min} are the maximum and minimum values of the k^{th} attribute, respectively, and $\overline{s}$ denotes a set of services that run on VMs.

3.3.2 QoS utility function of network

As part of our design of the QoS aggregation function for sequential service compositions, our calculating the QoS utility values of a fat-tree cloud datacenter network is done by considering mainly three quality criteria of a network: network delay, network throughput, and network reliability .

- **Network delay**. In this paper, the network delay is the sum of processing time for all switches for service composition:

$$nd(S)=nd_{core}+nd_{agg}+nd_{edge} \quad (10)$$

where $nd(S)$ represents the network delay of the service composition and $nd_{core}/nd_{agg}/nd_{edge}$ denote the total processing time of a concrete service (of service class $s_i(1\le i\le n)$) transferred by the core/aggregation/edge switches, respectively. They can be calculated as follows:

$$nd_{core}=\sum_{j=1,s_i\in S}^{c} time(core_j) \quad (11)$$

$$nd_{agg}=\sum_{j=1,s_i\in S}^{a} time(agg_j) \quad (12)$$

$$nd_{edge}=\sum_{j=1,s_i\in S}^{e} time(edge_j) \quad (13)$$

where $time(core_j)$ denotes the processing time of the j^{th} core switch, which is an average observation of past processing times of all core switches; $time(agg_j)$ denotes the processing time of the j^{th} aggregation switch, which is an average observation of past processing times for all aggregation switches; $time(edge_j)$ denotes the processing time of the j^{th} edge switch, which is an average observation of past processing times of all edge switches; c denotes the number of core switches; a denotes the number of aggregation switches; and e denotes the number of edge switches.

- **Network throughput**. Network throughput refers to the minimal average data rate of successful data delivery over a fat-tree cloud datacenter network for service composition, which is measured in bits per second (bps) as follows:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2603504, IEEE Transactions on Cloud Computing

$$nt(S)=\min\{nt_{core}, nt_{agg}, nt_{edge}\} \quad (14)$$

where $nt(S)$ represents the network throughput of service composition S, and $nt_{core} / nt_{agg} / nt_{edge}$ denote the minimal throughput of the core/aggregation/edge switches that processed a concrete service of service class $s_i (1 \le i \le n)$, which can be calculated as follows:

$$ns_{core} = \min_{s_i \in S}{}_{j=1}^{c} \{thou(core_j)\} \quad (15)$$

$$ns_{agg} = \min_{s_i \in S}{}_{j=1}^{a} \{thou(agg_j)\} \quad (16)$$

$$ns_{edge} = \min_{s_i \in S}{}_{j=1}^{e} \{thou(edge_j)\} \quad (17)$$

where $thou(core_j)$ denotes the throughput of the j^{th} core switch, which is an average observation of past throughputs of all core switches; $thou(agg_j)$ denotes the throughput of the j^{th} aggregation switch, which is an average observation of past throughputs of all aggregation switches; $thou(edge_j)$ denotes the throughput of the j^{th} edge switch, which is an average observation of past throughput of all edge switches; c denotes the number of core switches; a denotes the number of aggregation switches; and e denotes the number of edge switches.

- **Network reliability**. Network reliability is the product of the probabilities that a service request is correctly responded to within the maximum expected time frame (which is published in the switch description) for all witches. The value of network reliability is computed from historical data about past invocations using the following formula:

$$nr(S) = nr_{core} \times nr_{agg} \times nr_{edge} \quad (18)$$

where $nr(S)$ represents the network reliability of service composition S, and $nr_{core} / nr_{agg} / nr_{edge}$ denote the network reliability of a concrete service (of service class $s_i (1 \le i \le n)$) transferred by the core/aggregation/edge switches, which can be calculated as follows:

$$nr_{core} = \prod_{j=1, s_i \in S}^{c} reli(core_j) \quad (19)$$

$$nr_{agg} = \prod_{j=1, s_i \in S}^{a} reli(agg_j) \quad (20)$$

$$nr_{edge} = \prod_{j=1, s_i \in S}^{e} reli(edge_j) \quad (21)$$

where $reli(core_j)$ denotes the network reliability of the j^{th} core switch, which is an average observation of past reliability for all core switches; $reli(agg_j)$ denotes the network reliability of the j^{th} aggregation switch, which is an average observation of past reliability of all aggregation switches; $reli(edge_j)$ denotes the network reliability of the j^{th} edge switch, which is an average observation of past reliability for all edge switches; c denotes the number of core switches; a denotes the number of aggregation switches; and e denotes the number of edge switches.

Definition 3 (***QoS utility function of network***). The QoS utility function of a network for service composition is the QoS aggregation of all switches in a fat-tree cloud datacenter. Suppose that there are r QoS attributes of a network. The QoS utility function of a network for service composition S is defined as follows:

$$UN(S)=\sum_{k=1}^{r} \frac{QN_k^{max} - qn_k(S)}{QN_k^{max} - QN_k^{min}} . \omega_k, \quad (22)$$

with

$$\begin{cases} QN_k^{max} = \max_i qn_k(S_i) \\ QN_k^{\min} = \min_i qn_k(S_i) \end{cases}, \quad (23)$$

where $w_k \in R^+ (\sum_{k=1}^{r} w_k = 1)$ represents the weight of each QoS attribute of a network, $qn_k(S)$ denotes the k^{th} QoS attributes of a network (in this paper, $qn(S)=\{nd(S), nt(S), nr(S)\}$, and r=3); QN_k^{max} is the maximum value of the k^{th} attribute of service composition, and QN_k^{min} is the minimum value.

3.4 Problem Statement

The problem of finding the best service composition from all possible combinations is essentially an optimization problem in which the overall QoS utility value must be maximized and network resource consumption is a minimum while also satisfying global QoS constraints (which represent the user's end-to-end QoS requirements and can be expressed in terms of upper and/or lower bounds for the aggregated QoS values of a service composition) [36,39-41]. Via the terms and notations defined in Section 3.1, the optimization problem we address can be formalized as follows:

In a fat-tree cloud datacenter network environment, for a given service composition request $S = \{ s_1 , s_2 , \dots , s_m \}$ and a given set of r global QoS constraints $C = \{ c_1 , c_2 , \dots , c_r \}$, find a service composition solution for the cloud by binding each service class to a concrete service such that:

- The overall network resource consumption $NRS(S)$ is minimized,
- The overall QoS utility $U(S)+UN(S)$ is maximized, and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2603504, IEEE Transactions on Cloud Computing

- The aggregated QoS values of a composite service satisfy: $q(S)+qn(S) \le or \ge c_k, \forall c_k \in C, 1 \le k \le r$ (Note: In this paper, we choose $q(S)+qn(S) \le c_k$).

3.5 Finding the Optimal Solution

In this paper, the objective of cloud service composition is to reduce network resource consumption and deliver realistic QoS optimality while also satisfying global QoS constraints. Thus, network-aware service composition in the cloud can be formulated as a multi-objective constraint optimization problem, i.e., a maximization problem for overall QoS utility with a minimization problem for overall network resource consumption given by

$$\text{Min} \sum_{i=1, s_i \in S}^{m} (Packet_{core,i} + Packet_{agg,i} + Packet_{edge,i}), \quad (24)$$

$$\text{Max} \sum_{k=1}^{r} \left(\frac{Q_k^{\max} - q_k(S)}{Q_k^{\max} - Q_k^{\min}} + \frac{QN_k^{max} - qn_k(S)}{QN_k^{max} - QN_k^{min}} \right) . \omega_k , \quad (25)$$

subject to global QoS constraints as

$$q_k(S) + qn_k(S) \le c_k , \forall c_k \in C , \quad (26)$$

where n is the number of candidate services, m is the number of service classes, $Packet_{core,i}$ / $Packet_{agg,i}$ / $Packet_{edge,i}$ denote the total sizes of packets transferred by all switches for the concrete service from the i^{th} service class, $w_k \in R^+ (\sum_{k=1}^{r} w_k = 1)$ represents the weight of each QoS attribute, r denotes the number of QoS attributes, Q_k^{max} and Q_k^{min} are the maximum and minimum values of the k^{th} service QoS attribute, $q_k(S)$ denotes the aggregated service QoS value of the k^{th} service QoS attribute, QN_k^{max} and QN_k^{min} are the maximum and minimum values of the k^{th} network QoS attribute in the service composition, $qN_k(S)$ denotes the aggregated network QoS value of the k^{th} service QoS attribute, C denotes a given set of global QoS constraints, and c_k denotes the value of global QoS constraints, respectively.

For the problem, we aim at achieving two objectives simultaneously: minimizing the network resource consumption and maximizing the QoS utility value. This makes it feasible to find an optimal composition, as the number of candidate services and service classes is very small in the cloud. Hence, we adopt the mixed integer programming (MIP) method to solve the optimization problem of cloud service composition. The MIP method has been used to solve service composition problems by other researchers [13,40]. By solving (24-27) using any MIP solver method, a list of the best service candidates is obtained and returned to the service composition engine (or service broker) in the cloud.

4 Performance Evaluation

We implemented our approach in the credible cloud simulation system WebCloudSim [17,41], and comparatively evaluated our approach against several representative cloud service composition approaches in terms of network resource consumption, QoS optimality, and computation time under various service selection workloads and fat-tree network topology settings. Extensive experimental evaluation results show that our proposed approach, named NetMIP, is the best one in terms of effectiveness and efficiency.

4.1 WebCloudSim System

In order to evaluate our approach, we developed the WebCloudSim system and constructed a fat-tree cloud data center network environment, including core switches, aggregation switches, edge switches, network topology, physical servers, virtual machines, and services.

As shown in Fig. 2, when users input service composition requirements with global QoS constraints into the WebCloudSim system, the web server assigns them to the service composition server in use. The service composition server then adopts the service composition approach under evaluation to find the best services from the service/application servers. Conventional service composition approaches often select the best services in terms of their respective QoS properties. In contrast, our NetMIP approach searches for the best services and compose them into a composite service based upon not only their respective QoS properties, but also network resource consumption and network QoS measured and provided by the WebCloudSim server. Thus, compared with other approaches, our service composition process not only assures higher realistic QoS, but also effectively reduces network resource consumption in the cloud. More details on the WebCloudSim open source tool[2] are available from our previous work [38,42].

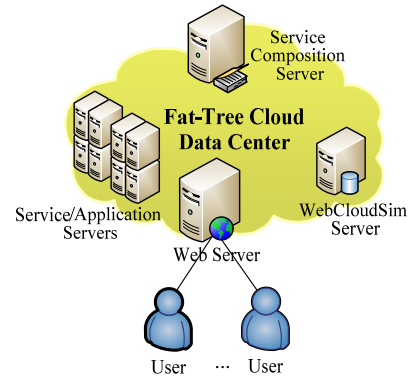


Fig. 2. Overview of the WebCloudSim system.

[2] www.webcloudsim.org

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2603504, IEEE Transactions on Cloud Computing

4.2 Experimental Evaluation Setup

In the WebCloudSim system, we established a 16-port fat-tree cloud datacenter consisting of 64 core switches and 16 pods. For the experimental evaluation setup, each pod comprised of 8 aggregation switches and 8 edge switches in a fat-tree cloud data center. That is, there were 128 aggregation switches and 128 edge switches. The bandwidth of the core and aggregation switches was set as 10 Gps and the bandwidth of the edge switch was set as 1 Gps. Each edge switch could connect to 8 physical servers. In the cloud datacenter, these physical servers were divided into two categories: 1) HP ProLiant G4 with one 3720 MIPS CPU and 4 GB memory, and 2) HP ProLiant G5 with one 5320 MIPS CPU and 4 GB memory.

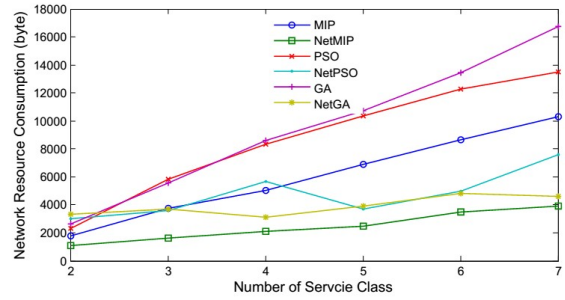
Each physical server could host 4 VMs, and each VM could host 2 services. Therefore, the datacenter contained 1024 host servers, 4096 VMs, and 8192 services. The base system configuration for the VM was 769 MB with 5.3 MB RAM disk, 1.6 MB kernel, 512 MB memory, and 1 GB disk. There were 8192 services in the cloud datacenter and each service had three QoS attributes: delay, throughput, and reliability. Since the target system was a fat-tree cloud datacenter network environment, it was extremely difficult to conduct repeatable, large-scale experiments on a real service and network infrastructure for the needed comparative evaluations. Hence, we used a collection of real QoS datasets [43-46] as the QoS data for all 8192 services to ensure the repeatability of experiments. The WebCloudSim system was configured to create 1,000 service composition requirements that were randomly executed while running the experiments.

4.3 Approaches

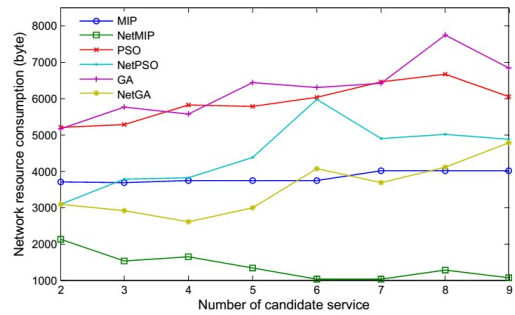
We compared the performance of NetMIP with five other approaches in terms of network resource consumption, QoS optimality, and computation time: PSO, NetPSO, GA, NetGA, and MIP.

- **PSO**. Particle swarm optimization (PSO) is often used to determine the optimal service composition in web or cloud environments [47,48], which optimizes a problem by moving particles around in the search space according to simple mathematical formulae involving the particles' positions and velocities. The initial population of particles was set to 20, and the maximum number of iterations was set to 30. We supplied this PSO with the standard QoS utility function, which does not model network QoS or network resource consumption. This is currently a standard approach to solving the service composition problem.
- **NetPSO**. It is a network-aware extension of PSO. Besides the same settings used for PSO, it is extended with our network QoS utility function and network resource consumption model, allowing it to determine realistic QoS optimality with low network resource consumption for cloud service composition.
- **GA**. The normal genetic algorithm (GA) uses uniform crossover and uniform mutation. The initial population for the GA was set to 20, and the maximum number of iterations was set to 30. We supplied this GA with the standard QoS utility function. This is also a standard approach [49,50] to solving the service composition problem presently.
- **NetGA**. It is a network-aware extension of GA. Besides the same settings used for GA, it is extended with our network QoS utility function and network resource consumption model.
- **MIP**. Using a mixed integer program (MIP) is one of the most common approaches [13,40] to solving the service composition problem in web or cloud environments. We supplied this MIP with the standard QoS utility function.
- **NetMIP**. Our proposed network-aware approach as described in Section 3.

All experiments were conducted on the same cloud datacenter established in the WebCloudSim cloud simulation system. A sufficient number of repetition tests were executed.



(a) Network resource consumption for service composition in the cloud with respect to the number of service classes.



(b) Network resource consumption for service composition in the cloud with respect to the number of candidate services.

Fig. 3 Comparison of network resource consumption for service composition in the cloud. Compared with MIP, GA, and PSO, our NetMIP method saves approximately 70% on average in network resources regardless of the number of service classes or candidate services in use. Compared with NetGA and NetPSO, our NetMIP saves approximately 50% on average in network resources regardless of the number of service classes or candidate services in use.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2603504, IEEE Transactions on Cloud Computing

4.4 Network Resource Consumption

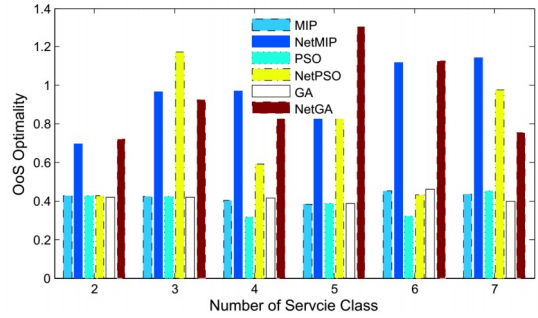
In the cloud, a composite service often contains several concrete services. We note that the number of concrete services of a composite service is generally less than 10 in practice [51]. Hence, in this experiment, the number of service classes used varied from 2 to 7, and the number of candidate services varied from 2 to 9. For example, when the number of service class is 2 and the number of candidate services also is 2, there are at most $1000*2^2$ services are used in the service composition process (note that many services are used multiple times). We set the delay of the composite service as the end-to-end QoS constraint and the size of network packet was set as 800 bytes. Fig. 3 illustrates the comparison of network resource consumption for the aforementioned six approaches.

Fig. 3 shows that our NetMIP enabled CSPs to save more network resources than other approaches. For example, in Fig. 3(a), the network resource consumption of our NetMIP approach is about 2,453 bytes on average, but other approaches (e.g., NetGA, at about 3,913 bytes, and GA, at about 9,624 bytes) are much more expensive in this respect. MIP, NetMIP, PSO, NetPSO, GA, and NetGA have network resource consumption ratios of about 3.0, 1.0, 4.0, 2.0, 4.0, and 2.0, respectively. When there is only one service composition requirement, our NetMIP approach can save more than 1 MB of network resources. When there is a large number of users who use service composition in the cloud, NetMIP could save a significant amount of network resources. Similarly, in Fig. 3(b), the network resource consumption of our NetMIP is about 1,379 bytes on average, but other approaches (e.g., NetGA, at about 3,536 bytes, and GA, at about 6,289 bytes) consume much more network resources on average. MIP, NetMIP, PSO, NetPSO, GA, and NetGA have network resource consumption ratios of about 3.0, 1.0, 4.0, 3.0, 5.0, and 3.0, respectively. With reference to Fig. 3, compared with MIP, GA, and PSO, our NetMIP approach saves approximately 70% on average in network resources regardless of the number of service classes or candidate services in use. Compared with NetGA and NetPSO, our NetMIP approach saves approximately 50% on average in network resources regardless of the number of service classes or candidate services in use. This means that our NetMIP approach can significantly reduce network resource consumption, and is the best among all of the evaluated approaches.

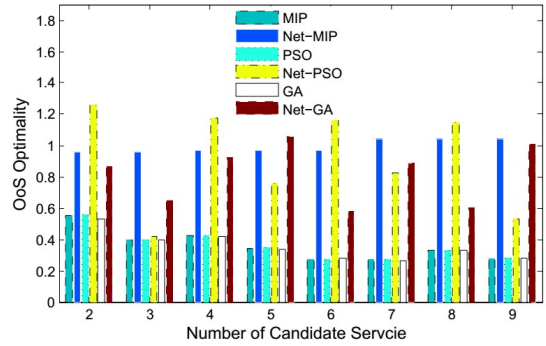
In contrast to MIP, PSO, and GA, our NetMIP approach considers the topology of fat-tree cloud datacenter networks, making full use of network QoS utility and a network resource consumption model to find the optimal solution for network-aware service composition in the cloud. In contrast to NetPSO and NetGA, our NetMIP approach uses mixed integer programming to optimize the cloud service composition problem. Owing to fat-tree topology, NetMIP is more effective than NetGA and NetMIP, which may be trapped in local optima.

In summary, NetMIP significantly reduces network resource consumption regardless of the number of service classes or candidate services used in solving the service composition problem in the cloud.

4.5 QoS Optimality



(a) OoS optimality of service composition in the cloud with respect to the number of service classes.



(b) QoS optimality of service composition in the cloud with respect to the number of candidate services.

Fig. 4 Comparison of QoS optimality of service composition in the cloud. Compared with MIP, PSO, and GA, NetMIP improves by approximately 50% on average in QoS optimality regardless of the number of service classes or candidate services in use. Compared with NetPSO and NetGA, NetMIP improves by approximately 15% on average in QoS optimality regardless of the number of service classes or candidate services in use.

In the QoS optimality experiment, the settings used were the same as those stated in Section 4.4. Fig. 4 illustrates the comparative evaluation results for QoS optimality, including network QoS. QoS optimality means that overall QoS utility is maximized.

Fig. 4 shows that NetMIP can significantly improve the QoS optimality of service composition in the cloud. For example, in Fig. 4(a), the QoS optimality of NetMIP is about 1.0 on average, but other approaches attained lower values on average. MIP, NetMIP, PSO, NetPSO, GA, and NetGA have network resource consumption ratios of about 0.4, 1, 0.4, 0.8, 0.4, and 0.9, respectively. Similarly, Fig. 4(b) shows that other approaches are also less effective than NetMIP on average. MIP, NetMIP, PSO, NetPSO, GA, and NetGA have network resource consumption ratios of about 0.4, 1.0, 0.4, 0.9, 0.4, and 0.8, respectively. With reference to Fig. 3, compared with MIP, PSO, and GA, NetMIP improves by approximately 50% on average in QoS opti-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2603504, IEEE Transactions on Cloud Computing

mality regardless of the number of service classes or candidate services in use. Compared with NetPSO and NetGA, NetMIP improves by approximately 15% on average in QoS optimality regardless of the number of service classes or candidate services in use. This means that NetMIP can effectively assure QoS optimality, and is the best among all of the evaluated approaches.

In contrast to MIP, PSO, and GA, NetMIP considers the network QoS when searching for the optimal solution for network-aware service composition in the cloud. This is different from NetPSO and NetGA because, in practice, the solution space of service composition problems in the cloud is not very large; our NetMIP approach is more effective than NetGA and NetMIP in QoS optimality in the context of fat-tree cloud datacenter networks.

Our NetMIP method significantly assures realistic QoS optimality regardless of the number of service classes or candidate services used when solving service composition problems in the cloud.

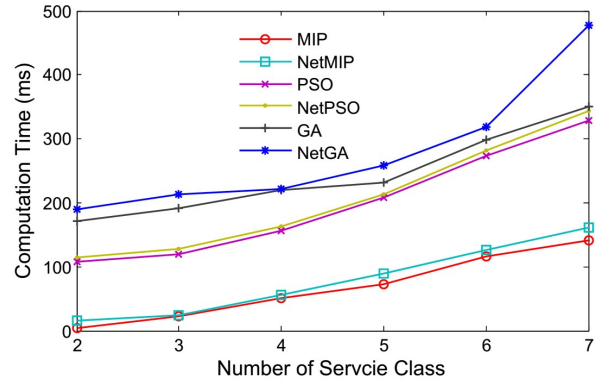
4.6 Computation Time

As shown in Fig. 5, we found that the computation time for NetMIP was very low. While the computation time for most approaches was higher than 200 ms, the computation time for NetMIP was about 80 ms on average regardless of the number of service classes or candidate services in use. With the exception of MIP, the computation time ratio between every other approach and NetMIP is about 2:1. Moreover, NetMIP incurs little or insignificant computation time overhead to the cloud as far as we can observe in our experiments, and yields more practical values for fat-tree cloud datacenters.

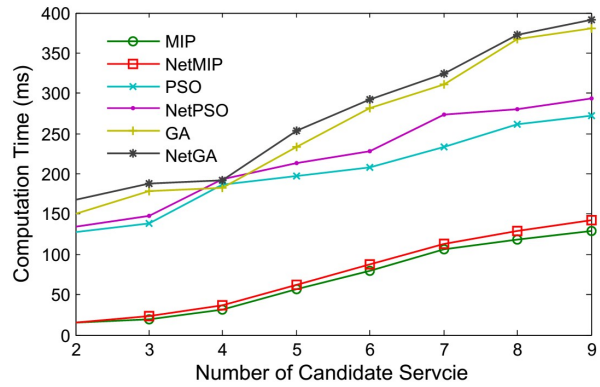
Finally, as Figs. 3, 4, and 5 show, our approach could achieve the best solutions with low network resource consumption, and has a roughly linear algorithmic complexity with respect to problem size. NetMIP outperforms all of the other representative approaches. This means that cloud service providers can adopt our approach to optimize cloud service (i.e., SaaS and IaaS) provision by a central control with all level information of services and network in the cloud data center.

4.7 Study of Parameters

In this section, for all of the evaluated approaches, we illustrate the effects of different parameter settings on network resource consumption, QoS optimality, and computation time. As shown in Figs. 6-8, the parameters include the bandwidth of the core and aggregation switches, the bandwidth of edge switches and the number (i.e., the parameter *ps*) of physical servers connected to one edge switch.

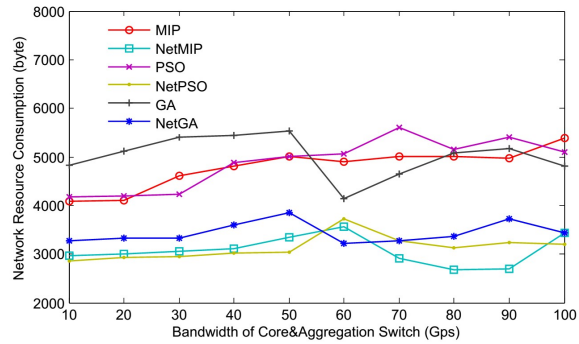


(a) Computation time for service composition in the cloud with respect to the number of service classes.

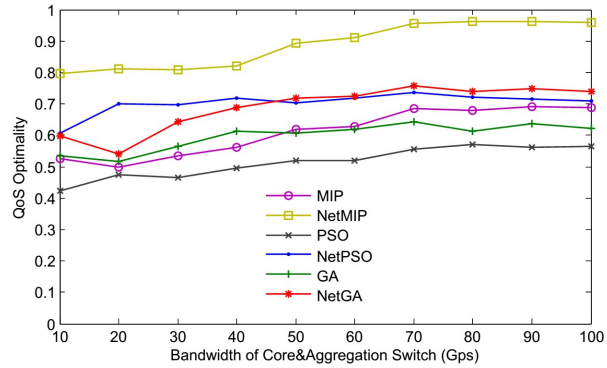


(b) Computation time for service composition in the cloud with respect to the number of candidate services.

Fig. 5 Comparison of computation time for service composition in the cloud. MIP and NetMIP are the fastest methods, with computation times lower than 150 ms regardless of the number of service classes or candidate services in use.

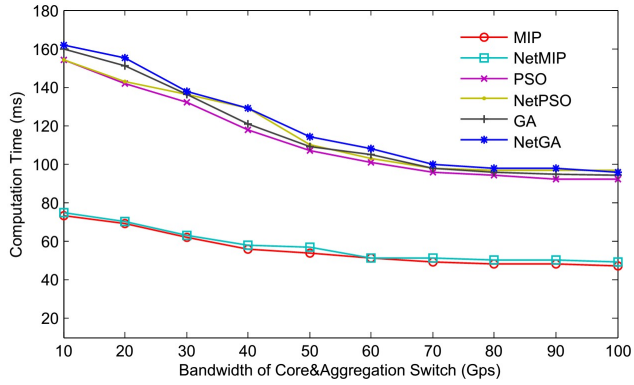


(a) Effect of the bandwidth of the core and aggregation switches on network resource consumption



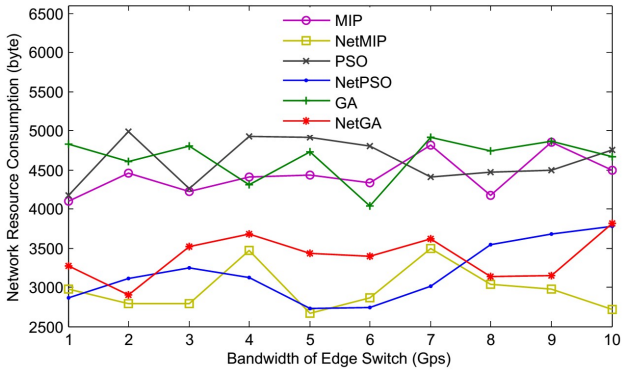
(b) Effect of the bandwidth of the core and aggregation switches on QoS optimality

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2603504, IEEE Transactions on Cloud Computing

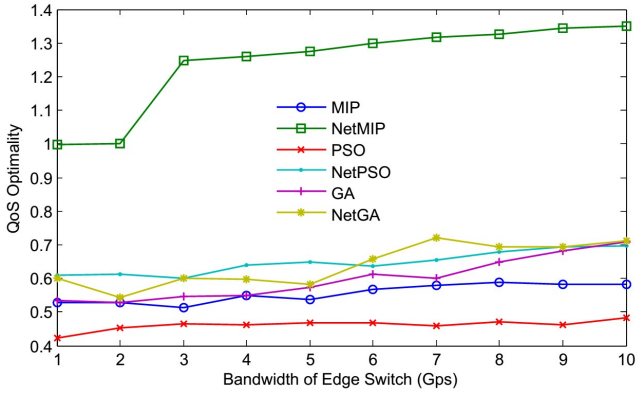


(c) Effect of the bandwidth of the core and aggregation switches on computation time

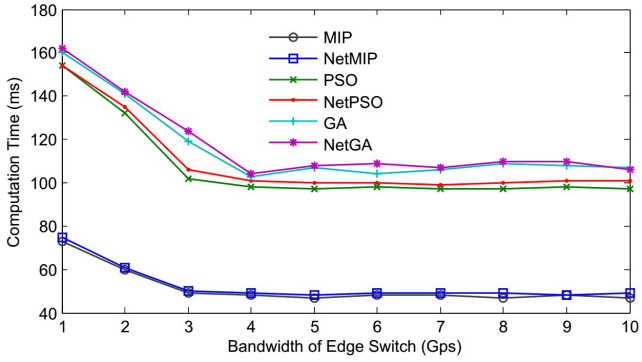
Fig. 6 Effect of the bandwidth of the core and aggregation switches on the network resource consumption, QoS optimality, and computation time.



(a) Effect of the bandwidth of the edge switch on network resource consumption

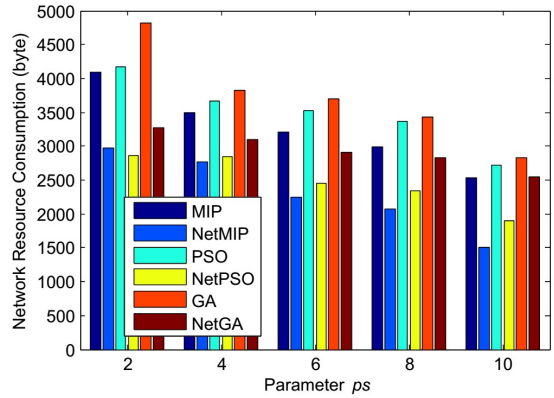


(b) Effect of the bandwidth of the edge switch on QoS optimality

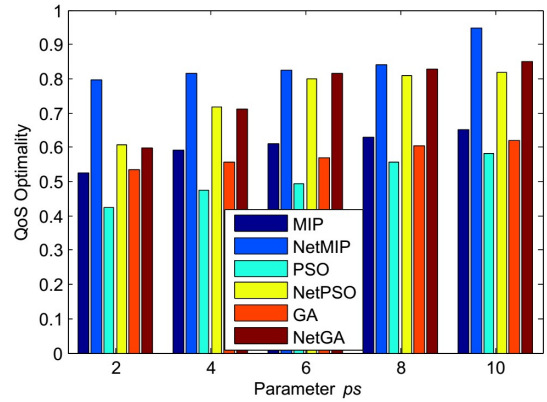


(c) Effect of the bandwidth of the edge switch on computation time

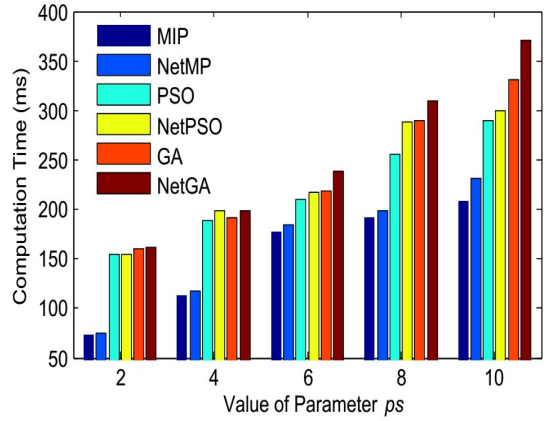
Fig. 7 Effect of the bandwidth of the edge switch on network resource consumption, QoS optimality, and computation time.



(a) Effect of the parameter *ps* on network resource consumption



(b) Effect of the parameter *ps* on QoS optimality



(c) Effect of the parameter *ps* on computation time

Fig. 8 Effect of the parameter *ps* on network resource consumption, QoS optimality, and computation time

4.7.1 Effect of the bandwidth of the core and aggregation switches

Figs. 6 (a), (b), and (c) show the effect of the bandwidth of the core and aggregation switches on network resource consumption, QoS optimality, and computation time. To clearly show its impact, we varied the value of bandwidth from 10 Gps to 100 Gps with a step value of 10 Gps. The bandwidth of the edge switch was set to 1 Gps. Other settings were the same as stated in Section 4.2.

In short, Fig. 6 shows: 1) our NetMIP approach is the best among all of the evaluated approaches in terms of network resource consumption and QoS optimality; 2) the computation time of NetMIP is very close to that of MIP,

and both values are much lower than that of other approaches; 3) the network resource consumption of NetMIP was not substantially influenced by increasing bandwidth, and its network resource consumption was the lowest on average; 4) the less the computation time of NetMIP, the higher the bandwidth of the core and aggregation switches; 5) the better the QoS optimality of NetMIP, the higher the bandwidth of the core and aggregation switches.

4.7.2 Effect of the bandwidth of the edge switch

Figs. 7 (a), (b), and (c) show the effect of the bandwidth of the edge switch on network resource consumption, QoS optimality, and computation time. To clearly show its impact, we varied the value of bandwidth from 1 Gps to 10 Gps with a step value of 1 Gps. The bandwidth of the core and aggregation switches was set to 10 Gps. Other settings were the same as stated in Section 4.7.1.

In short, Fig. 7 shows: 1) NetMIP is the best among all of the evaluated approaches in terms of network resource consumption and QoS optimality; 2) the computation time of NetMIP was very close to that of MIP, and both were much lower than that of other approaches; 3) the network resource consumption of NetMIP was not substantially influenced by increasing bandwidth of the edge switch, and its network resource consumption was still the lowest on average; 4) the lower the computation time of NetMIP, the higher the bandwidth of the edge switch; 5) the better the QoS optimality of NetMIP, the higher the bandwidth of the edge switch.

4.7.2 Effect of the parameter ps

Figs. 8 (a), (b), and (c) show the effect of the number of physical servers connected to one edge switch on network resource consumption, QoS optimality, and computation time. To clearly show its impact, we varied the value of the parameter *ps* from 2 to 10 with a step value of 2. The bandwidth of the core and aggregation switches was set as 10 Gps. The bandwidth of the edge witch was set as 1 Gps. Other settings were the same as stated in Section 4.7.2.

In short, Fig. 8 shows: 1) NetMIP is the best among all of the evaluated approaches in terms of network resource consumption and QoS optimality; 2) the computation time of NetMIP was very close to that of MIP, and both values are much lower than that of other approaches; 3) the lower the network resource consumption of NetMIP, the greater the number of physical severs; 4) the network resource consumption of NetMIP was still the lowest on average; 5) the greater the computation time of our NetMIP method, the higher the number of physical severs connected to one edge switch; 6) the better the QoS optimality of NetMIP, the greater the number of physical servers.

5 Conclusions

In this paper, we proposed a network-aware approach to service composition in a fat-tree cloud datacenter, consisting of a network resource consumption function, a network QoS computation, and a network-aware service composition model. In contrast to current approaches in this area, our approach links services and networks, and considers network resource consumption in the service composition process. It has a roughly linear computation time with respect to problem size. Via the credible WebCloudSim system, we have validated that our NetMIP approach outperforms several representative cloud service composition approaches in terms of network resource consumption, QoS optimality, and computation time.

Note that our approach is not suitable for multiple data centers due to network topology confliction or non-disruptive load balancing strategy. In the future, we would like to investigate how our approach fares in scenarios with other data center sizes and more QoS attributes. We would also like to evaluate our approach based on other cloud datacenter network topologies and explore a good means of enhancing our proposed approach with software defined network technologies.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation of China (61472047 and 61272521).

REFERENCES

[1] W. Yi and M. B. Blake, "Service-Oriented Computing and Cloud Computing: Challenges and Opportunities," *IEEE Internet Computing*, 6, vol. 14, pp. 72-75, 2010.

[2] A. Strunk, "QoS-Aware Service Composition: A Survey," *In Proceedings of IEEE 8th European Conference on Web Services (ECOWS 2010)*, 2010, pp. 67-74.

[3] R. Mietzner, C. Fehling, D. Karastoyanova, and F. Leymann, "Combining horizontal and vertical composition of services," *In Proceedings of IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2010)*, 2010, 2010, pp. 1-8.

[4] M. P. Papazoglou and D. Georgakopoulos, "Introduction: Service-oriented computing," *Communications of the ACM*, 10, vol. 46, pp. 24-28, 2003.

[5] A. Goscinski and M. Brock, "Toward dynamic and attribute based publication, discovery and selection for cloud computing," *Future Generation Computer Systems*, 7, vol. 26, pp. 947-970, 2010.

[6] M. Brock and A. Goscinski, "Toward ease of discovery, selection and use of clusters within a cloud," *In Proceedings of the 3rd IEEE International Conference on Cloud Computing (CLOUD 2010)*, 2010, pp. 289-296.

[7] W. Shangguang, Z. Zheng, S. Qibo, Z. Hua, and Y. Fangchun, "Cloud model for service selection," *In Proceedings of the 30th IEEE Conference on Computer Communications Workshops on Cloud Computing (INFOCOM WKSHPS 2011)*, 2011, pp. 666-671.

[8] R. Karim, D. Chen, and A. Miri, "End-to-End QoS Prediction of Vertical Service Composition in the Cloud," *In Proceedings of IEEE 8th International Conference on Cloud Computing (CLOUD 2015)*, 2015, pp. 229-236.

[9] A. V. Dastjerdi and R. Buyya, "Compatibility-Aware Cloud Service Composition under Fuzzy Preferences of Users," *IEEE*

Transactions on Cloud Computing, 1, vol. 2, pp. 1-13, 2014.

[10] K. Kritikos and D. Plexousakis, "Multi-cloud Application Design through Cloud Service Composition," *In Proceedings of IEEE 8th International Conference on Cloud Computing (CLOUD 2015)*, 2015, pp. 686-693.

[11] S.-Y. Hwang, E.-P. Lim, C.-H. Lee, and C.-H. Chen, "Dynamic Web service selection for reliable Web service composition," *IEEE Transactions on Services Computing*, 2, vol. 1, pp. 104-116, 2008.

[12] M. Alrifai, T. Risse, and W. Nejdl, "A Hybrid Approach for Efficient Web Service Composition with End-to-End QoS Constraints," *ACM Transactions on the Web*, 2, vol. 6, pp. 1-31, May 2012.

[13] 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, vol. 1, pp. 1-26, 2007.

[14] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality driven web services composition," *In Proceedings of the 12th international conference on World Wide Web (WWW 2003)*, 2003, pp. 411-421.

[15] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Transactions on Software Engineering*, 5, vol. 30, pp. 311-327, 2004.

[16] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *In Proceedings of ACM SIGCOMM Conference on Data Communication (SIGCOMM 2008)*, 2008, pp. 63-74.

[17] D. Jun, G. Zhiyang, and Y. Yuanyuan, "Cost efficient and performance guaranteed virtual network embedding in multicast fat-tree DCNs," *In Proceedings of IEEE Conference on Computer Communications (INFOCOM 2015)*, 2015, pp. 136-144.

[18] D. Breitgand and A. Epstein, "Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds," *In Proceedings of IEEE Conference on Computer Communications (INFOCOM 2012)*, 2012, pp. 2861-2865.

[19] A. Jula, E. Sundararajan, and Z. Othman, "Cloud computing service composition: A systematic literature review," *Expert Systems with Applications*, 8, vol. 41, pp. 3809-3824, 2014.

[20] M. J. Buco, R. N. Chang, L. Z. Luan, C. Ward, J. L. Wolf, and P. S. Yu, "Utility computing SLA management based upon business objectives," *IBM Systems Journal*, 1, vol. 43, pp. 159-178, 2004.

[21] J. O. Gutierrez-Garcia and K.-M. Sim, "Self-Organizing Agents for Service Composition in Cloud Computing," *In Proceedings of the IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom 2010)*, 2010.pp. 59-66.

[22] J. O. Gutierrez-Garcia and K. Sim, "Agent-based Cloud service composition," Applied Intelligence, 3, vol. 38, pp. 436-464.

[23] C. Zeng, X. Guo, W. Ou, and D. Han, "Cloud Computing Service Composition and Search Based on Semantic," *Cloud Computing*, Springer Berlin Heidelberg, 2009, pp. 290-300.

[24] T. Wei-Tek, Z. Peide, J. Balasooriya, C. Yinong, B. Xiaoying, and J. Elston, "An Approach for Service Composition and Testing for Cloud Computing," *In Proceedings of the 10th International Symposium on Autonomous Decentralized Systems (ISADS 2011)*, 2011, pp. 631-636.

[25] H. Cai and L. Cui, "Cloud Service Composition Based on Multi-Granularity Clustering," *Journal of Algorithms & Computational Technology*, 2, vol. 8, pp. 143-162, 2014.

[26] F. Alrebeish and R. Bahsoon, "Stabilising Performance in Cloud Services Composition Using Portfolio Theory," *In Proceedings of IEEE International Conference on Web Services (ICWS 2015)*, 2015, pp. 1-8.

[27] S. Sundareswaran, A. Squicciarini, and D. Lin, "A brokerage-based approach for cloud service selection," *In Proceeding of IEEE 5th International Conference on Cloud Computing (CLOUD 2012),* 2012, pp. 558-565.

[28] W. Zeng, Y. Zhao, and J. Zeng, "Cloud service and service selection algorithm research," *In Proceedings of ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC 2009)*, 2009, pp. 1045-1048.

[29] H. Qiang, H. Jun, C. Feifei, W. Yanchun, R. Vasa, Y. Yun, and J. Hai, "QoS-Aware Service Selection for Customisable Multi-tenant Service-Based Systems: Maturity and Approaches," *In Proceedings of IEEE 8th International Conference on in Cloud Computing (CLOUD 2015)*, 2015, pp. 237-244.

[30] E. Wittern, J. Kuhlenkamp, and M. Menzel, "Cloud Service Selection Based on Variability Modeling," Springer Berlin Heidelberg, 2012, pp. 127-141.

[31] L. Qu, Y. Wang, M. A. Orgun, L. Liu, H. Liu, and A. Bouguettaya, "CCCloud: Context-aware and credible cloud service selection based on subjective assessment and objective assessment," *IEEE Transactions on Services Computing*, 3, vol. 8, pp. 369-383, 2015.

[32] L. Qu, Y. Wang, M. A. Orgun, L. Liu, and A. Bouguettaya, "Context-aware cloud service selection based on comparison and aggregation of user subjective assessment and objective performance assessment," *In Proceedings of IEEE International Conference on Web Services (ICWS 2014)*, 2014, pp. 81-88.

[33] L. Sun, H. Dong, F. K. Hussain, O. K. Hussain, and E. Chang, "Cloud service selection: State-of-the-art and future research directions," *Journal of Network and Computer Applications*, 1, vol. 45, pp. 134-150, 2014.

[34] H. Jun, L. Guoquan, D. Qiang, and Y. Yuhong, "QoS-Aware Service Composition for Converged Network-Cloud Service Provisioning," *In Proceedings of IEEE International Conference on Services Computing (SCC 2014)*, 2014, pp. 67-74.

[35] F. B. Charrada and S. Tata, "An efficient algorithm for the bursting of service-based applications in hybrid Clouds," *IEEE Transactions on Services Computing*, DOI: 10.1109/TSC.2015.2396076.

[36] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions on Software Engineering*, 6, vol. 33, pp. 369-384, Jun 2007.

[37] J. H. Jang, D. H. Shin, and K. H. Lee, "Fast quality driven selection of composite Web services," *In Proceedings of 4th European Conference on Web Services (ECOWS 2006)*, 2006, pp. 87-96.

[38] A. Zhou, S. Wang, Z. Zheng, C. Hsu, M. Lyu, and F. Yang, "On Cloud Service Reliability Enhancement with Optimal Resource Usage," *IEEE Transactions on Cloud Computin*g, DOI: 10.1109/tcc.2014.2369421.

[39] A. Charfi and M. Mezini, "Middleware Services for Web Service Compositions," *In Proceedings of the 14th International conference on World Wide Web (WWW 2005)*, 2005, pp. 1132-1133.

[40] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient QoS-aware service composition," *In Proceedings of the 18th international conference on World Wide Web (WWW2009)*, 2009, pp. 881-890.

[41] S. G. Wang, Z. B. Zheng, Q. B. Sun, H. Zou, and F. C. Yang, "Reliable web service selection via QoS uncertainty computing," *International Journal of Web and Grid Service*s, 4, vol. 7, pp. 410-426, 2011.

[42] A. Zhou, S. Wang, Q. Sun, H. Zou, and F. Yang, "FTCloudSim: a simulation tool for cloud service reliability enhancement mechanisms," *In Proceedings of ACM/IFIP/USENIX International Middleware Conference (Middleware 2013)*, 2013, pp. 1-2.

[43] E. Al-Masri and Q. H. Mahmoud, "Discovering the best web service," *In Proceedings of the 16th International Conference on World Wide Web (WWW 2007),* 2007, pp. 1257-1258.

[44] E. Al-Masri and Q. H. Mahmoud, "QoS-based Discovery and Ranking of Web Services," *In Proceedings of 16th International Conference on Computer Communications and Networks (ICCCN 2007)*, 2007, pp. 529-534.

[45] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed QoS evaluation for real-world Web services," *In Proceedings of IEEE 8th International Conference on Web Services (ICWS 2010)*, 2010, pp. 83-90.

[46] Z. Yilei, Z. Zibin, and M. R. Lyu, "Exploring Latent Features for Memory-Based QoS Prediction in Cloud Computing," *In Proceedings of the 30th IEEE Symposium on in Reliable Distributed Systems (SRDS 2011)*, 2011, pp. 1-10.

[47] S. Wang, Q. Sun, H. Zou, and F. Yang, "Particle Swarm Optimization with Skyline Operator for Fast Cloud-based Web Service Composition," *Mobile Networks and Applications*, 1, vol. 18, pp. 116-121, 2013/02/01 2013.

[48] H. Jiwei and L. Chuang, "Agent-Based Green Web Service Selection and Dynamic Speed Scaling," In Proceedings of IEEE 20th International Conference on Web Services (ICWS 2013), 2013, pp. 91-98.

[49] P. Bartalos and M. B. Blake, "Engineering Energy-Aware Web Services toward Dynamically-Green Computing," In Proceedings of International Conference on Service-Oriented Computing Workshops (ICSOC 2012), 2012, pp. 87-96.
[50] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "An approach for QoS-aware service composition based on genetic algorithms," *In Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GEC 2005)*, 2005, pp. 1069-1075.
[51] J. Lin, B. Liu, N. Sadeh, and J. I. Hong, "Modeling Users' Mobile App Privacy Preferences: Restoring Usability in a Sea of Permission Settings," In Proceedings of Symposium on Usable Privacy and Security (SOUPS 2014), 2014, pp.199-212.

Shangguang Wang is an associate professor at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications (BUPT). He received his Ph.D. degree at BUPT in 2011. He is 2015-2016 President of the Service Society Young Scientist Forum in China, General Chair of CollaborateCom 2016, General Chair of ICCSA 2016, Application Track Chair of IEEE SCC 2015, Workshop Chair of AMC/IEEE UCC 2016, Program Chair of IOV 2014, and Program Chair of SC2 2014. He has published more than 50 journal/conference papers such as IEEE TSC, ACM TOMM, IEEE TCC, IEEE TETC, and IEEE TASE. His research interests include Service Computing, Cloud Services, and QoS Management. He is a Senior Member of the IEEE.

Ao Zhou is an assistant professor at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She received her Ph.D. degree in computer science at Beijing University of Posts and Telecommunications of China in 2015. Her research interests include cloud computing, service reliability.

Fangchun Yang received his Ph.D.in communications and electronic systems from the Beijing University of Posts and Telecommunications in 1990. He is currently a professor at the Beijing University of Posts and Telecommunication, China. He has published six books and more than 80 papers. His current research interests include network intelligence, service computing, communications software, soft-switching technology, and network security. He is a fellow of the IET.

Rong Chang received his PhD degree in computer science and engineering from the University of Michigan in 1990. He is with IBM Research leading a global team creating innovative IoT cloud services technologies. He holds 30+ patents and has published 40+ papers. He is Member of IBM Academy of Technology, ACM Distinguished Engineer, Chair of IEEE Computer Society Technical Committee on Services Computing, Editor-in-Chief of the International Journal of Cloud Computing and Associate Editor-in-Chief of the IEEE Transactions on Services Computing.