T-*Broker*: A Trust-Aware Service Brokering Scheme for Multiple Cloud Collaborative Services

Xiaoyong Li, Huadong Ma, *Member, IEEE*, Feng Zhou, and Wenbin Yao

***Abstract*— Oriented by requirement of trust management in multiple cloud environment, this paper presents T-*broker*, a trust-aware service brokering scheme for efficient matching cloud services (or resources) to satisfy various user requests. First, a trusted third party-based service brokering architecture is proposed for multiple cloud environment, in which the T-*broker* acts as a middleware for cloud trust management and service matching. Then, T-*broker* uses a hybrid and adaptive trust model to compute the overall trust degree of service resources, in which trust is defined as a fusion evaluation result from adaptively combining the direct monitored evidence with the social feedback of the service resources. More importantly, T-*broker* uses the maximizing deviation method to compute the direct experience based on multiple key trusted attributes of service resources, which can overcome the limitations of traditional trust schemes, in which the trusted attributes are weighted manually or subjectively. Finally, T-*broker* uses a lightweight feedback mechanism, which can effectively reduce networking risk and improve system efficiency. The experimental results show that, compared with the existing approaches, our T-*broker* yields very good results in many typical cases, and the proposed system is robust to deal with various numbers of dynamic service behavior from multiple cloud sites.**

***Index Terms*— Multiple cloud computing, trust-aware service brokering, resource matching, feedback aggregation.**

I. Introduction

MULTIPLE cloud theories and technologies are the hot directions in the cloud computing industry, which a lot of companies and government are putting much concern to make sure that they have benefited from this new innovation [1], [2]. However, compared with traditional networks, multiple cloud computing environment has many unique features such as resources belonging to each cloud provider, and such resources being completely distributed, heterogeneous, and totally virtualized; these features indicate that unmodified traditional trust mechanisms can no longer be used in multiple cloud computing environments. A lack of trust between cloud users and providers has hindered the universal acceptance of clouds as outsourced computing services [3], [4]. Thus, the development of trust awareness technology for cloud computing has become a key and urgent research direction [5]–[8]. Today, the problem of trusted cloud computing has become a paramount concern for most users. It's not that the users don't trust cloud computing's capabilities; rather, they mainly question the cloud computing's trustworthiness [9]–[11].

Manuscript received August 29, 2014; revised December 23, 2014; accepted February 25, 2015. Date of publication March 13, 2015; date of current version May 15, 2015. This work was supported in part by the National Nature Science Foundation of China under Grant 61370069, Grant 61320106006, and Grant 61332005, in part by the Fok Ying Tung Education Foundation under Grant 132032, in part by the Program for New Century Excellent Talents in University under Grant NCET-12-0794, in part by the Beijing Natural Science Foundation under Grant 4111002, and in part by the Cosponsored Project of Beijing Committee of Education. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Dinei A. Florencio.

X. Li is with the Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: lxyxjtu@163.com).

H. Ma, F. Zhou, and W. Yao are with the Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: mhd@bupt.edu.cn; zfeng@bupt.edu.cn; yaowenbincdc@163.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2015.2413386

A. Motivation

Recently, cloud brokering systems have emerged as a promising concept to offer enhanced service of cloud environment, such as RESERVOIR [18], PCMONS [19], RightScale [20], SpotCloud [21], Aeolus [22] and OPTIMIS [12]. These cloud brokers can provide intermediation and aggregation capabilities to enable users to deploy their virtual infrastructures across cloud systems. The future of cloud computing will be permeated with the emergence of cloud brokers acting as an intermediary between cloud providers and users to negotiate and allocate resources among multiple sites. Unfortunately, apart from OPTIMIS [12], most of these brokers do not provide trust management capabilities for multiple cloud collaborative computing, such as how to select the optimal cloud resources to deploy a service, how to optimally distribute the different components of a service among different clouds, or even when to move a given service component from a cloud to another to satisfy some optimization criteria.

From many scholars understanding [6]–[11], [13], [14], [17], to increase the adoption of cloud services, cloud providers must first establish trust to alleviate the worries of a large number of users [25]. To manage and schedule resources with high trustworthy, we need an accurate way of measuring and predicting usage patterns of computing resources whose patterns are changing dynamically overtime. From here, the main motivation of this paper is to construct a trust-aware service brokering system for efficient matching computing resources to satisfy various user requests.

Although several scholars have been attracted by this question and carried out some studies [7]–[9], [13], [14], [17], their methods have not been able to breakthrough the existing ideas in previous trust models [15], [16], [26]–[29]. First, some hybrid trust models are proposed for cloud computing environment (see [13], [14]). It is no doubt that how to adaptively fuse direct trust (first-hand trust) and indirect trust (users' feedback) should be an important problem, however, most current studies in hybrid trust models either ignore the problem or using subjective or manual methods to assign weight to this two trust factors (first-hand trust and users' feedback) (see [13], [14]). This may lead to misinformation and preclude an accurate evaluation of trustworthiness. At the same time, evidence-based trust evaluation can reflects real-time behavior of service providers [7], [9], and it should be a process of multi-attribute decision-making. Avoiding the effect of individual favouritism on the weight allocation of trust indicators is a key task. However, most previous studies used subjective methods to weight the trust indicators (see [7], [9]). Their approaches do not reflect trust decision-making adaptability, and may lead to deviation from objective facts. Furthermore, consider industry data centers, which host hundreds of machines and handles thousands of request per second, the delay induced by trust system can be one big problem. There is no doubt that the efficiency of a trust system is an important requirement for multiple cloud environment. That is, the trust brokering system should be fast convergence and light-weight to serve for a large number of users and providers. However, existing studies paid little attention to this question, which greatly affects scalability and availability of the trust system (see [14], [29]).

B. Our Contributions

Based on previous work on trust management [10], [11], [30], [31], [36], this paper presents T-*broker* for efficient matching computing resources to satisfy various user requests in the multi-cloud environment. The main innovations of our scheme go beyond those of existing approaches in terms of the following three aspects:

- T-*broker* uses a trust-aware brokering architecture, in which the broker itself acts as the TTP for trust management and resource scheduling. Through distributed soft-sensors, this brokering architecture can real-time monitor both dynamic service behavior of resource providers and feedbacks from users.
- T-*broker* uses a hybrid and adaptive trust model to compute the overall trust degree of service resources, in which trust is defined as a fusion evaluation result from adaptively combining dynamic service behavior with the social feedback of the service resources.
- T-*broker* uses a maximizing deviation method to compute the direct trust of service resource, which can overcome the limitations of traditional trust models, in which the trusted attributes are weighted manually or subjectively. At the same time, this method has a faster convergence than other existing approaches.

These innovative designs and other specific features (e.g., real-time trust degree calculation approach based on time attenuation function and lightweight feedback mechanism) collectively make T-*broker* an efficient solution that can be used in multi-cloud environment. The experimental results show that, compared with the existing approaches, our T-*broker* yields very good results in many typical cases, and the proposed system is robust to deal with various numbers of dynamic service behavior from multiple cloud sites.

The remaining parts of this paper are organized as follows: Section II gives an overview of related work. T-*broker*'s architecture is described in Section III. Section IV outlines the details of the trust calculation mechanism. The experimental results are presented in Section V. Finally, Section VI concludes the paper and suggests future directions.

II. Related Work

The main contributions of our trust scheme are based on many existing representative work. In this section, we first review the typical work of cloud brokers. We then analyze the developments of trust management in cloud computing.

A. Development of Cloud Brokers

In recent years, there are many cloud service brokers or monitoring systems emerged as a promising concept to offer enhanced service delivery over large-scale cloud environments. Some private companies offer brokering solutions for the current cloud market, e.g., RightScale [20] or SpotCloud [21].

In [18], the authors use the Lattice monitoring framework as a real-time feed for the management of a service cloud. Monitoring is a fundamental aspect of Future Internet elements, and in particular for service clouds, where it is used for both the infrastructure and service management. The authors present the issues relating to the management of service clouds, discussing the key design requirements and how these are addressed in the RESERVOIR project [18]. The authors also present the Lattice monitoring framework, discussing its main features and also giving an overview of its design and implementation, together with a presentation of its use within RESERVOIR.

In [19], the authors point out, although many solutions are now available, cloud management and monitoring technology has not kept pace, partially because of the lack of open source solutions. To address this limitation, the authors describe their experience with a private cloud, and discuss the design and implementation of a private cloud monitoring system (PCMONS) and its application via a case study for the proposed architecture. An important finding of this work is that it is possible to deploy a private cloud within the organization using only open source solutions and integrating with traditional tools like Nagios. However, there is significant development work to be done while integrating these tools.

RightScale [20] is a web based cloud computing managing tool for managing cloud infrastructure from multiple providers. RightScale enables organizations to easily deploy and manage business-critical applications across public, private, and hybrid clouds. SpotCloud [21] provides a structured cloud capacity marketplace where service providers sell the extra capacity they have and the buyers can take advantage of cheap rates selecting the best service provider at each moment. The broker in [24] also provides this feature but in an automatized way, without checking manually the prices of each cloud provider at each moment. Thus, optimization algorithms can be used to select the best way to place the VM according to the actual rates of the cloud providers.

Aeolus [22] is an open source cloud management software sponsored by Red Hat, which runs on Linux systems. It provides both ease the burden of managing large numbers of clouds, as well as ensure that cloud consumers can use large numbers of clouds to avoid getting locked into the offering of any single provider. Besides managing virtual machines in various clouds, a cross-cloud broker like Aeolus needs to be able to build images for these clouds from a single specification, track that images have been converted and uploaded into what cloud, as well as automate image updates. To further simplify the management of complicated cloud uses, Aeolus makes it possible to describe multi-instance applications like three-tier web applications as one unit, from image definition to upload and launch into target clouds. The main components of Aeolus are Conductor, the application that users and administrators interact with, Composer, an application and tools for building and managing images, and Orchestrator, tooling for treating groups of virtual machines as one application.

According to the Cloud Security Alliance's work, a cloud is modeled in seven layers: Facility, network, hardware, OS, middle ware, application, and the user. These layers can be controlled by either the cloud provider or the cloud customer. In [23], the author presents a set of recommended restrictions and audits to facilitate cloud security. The author found, although the recommendations might be overkill for deployments involving no sensitive data, they might be insufficient to allow certain information to be hosted in any public or community cloud.

EU project OPTIMIS has addressed the trust brokering problem for multiple clouds and a full-fledged OPTIMIS toolkit is available for the developers at present [12]. OPTIMIS can identify, capture and codify what an optimized cloud ecosystem driven by trust, risk, eco-efficiency and cost will look like. The OPTIMIS framework and toolkit will simplify service construction, and support deployment and runtime decisions based on prior evaluation of providers. OPTIMIS deliverables can enable clouds to be composed from multiple services and resources. It will support service brokerage via interoperability, and is architecture-independent.

B. Trust in Cloud Computing

Several research groups both in academia and industry are working in the area of trust management in cloud computing environment. This section will take an in-depth look at the recent developments in this area.

Khan et al. have reviewed the trust needs in the cloud system [5]. They analyze the issues of trust from what a cloud user would expect with respect to their data in terms of security and privacy. They further discuss that what kind of strategy the service providers may undertake to enhance the trust of the user in cloud services and providers. They have identified control, ownership, prevention and security as the key aspects that decide users' level of trust on services. Diminishing control and lack of transparency have identified as the issues that diminishes the user's trust on cloud systems. The authors have predicted that remote access control facilities for resources of the users, transparency with respect to cloud providers actions in the form of automatic traceability facilities, certification of cloud security properties and capabilities through an independent certification authority and providing security enclave for users could be used to enhance the trust of users in the services.

Hwang and Li suggested using a trust-overlay network over multiple data centers to implement a reputation system for establishing trust between service providers and data owners [6]. The authors build reputation systems using a Distributed-hash-table (DHT)-based trust-overlay networks among virtualized data centers and distributed file systems. These networks over cloud resources provisioned from multiple data centers for trust management and distributed security enforcement. Data coloring and software watermarking techniques protect shared data objects and massively distributed software modules. These techniques safeguard multi-way authentications, enable single sign-on in the cloud, and tighten access control for sensitive data in both public and private clouds. In [25], Hwang and Kulkarni also presented an integrated cloud architecture to reinforce the security and privacy in cloud applications. They propose an approach to integrating virtual clusters, security-reinforced datacenters, and trusted data accesses guided by reputation systems. However, in [6] and [25], the authors only focused on the trust and privacy issues of user-side, and they did not mention about server-side trust problem.

Kim et al. present a trust model for allocation of resources to satisfy various user requests [7]. Their trust model collects and analyzes reliability based on historical information of servers in a Cloud data center. Then their trust model prepares the best available resources for each service request in advance, and provide the best resources to users. In [7], the cloud trust model for cloud resource is defined as: $G_i = w_{Rc}Rc_i + w_{Ru}Ru_i + w_T T_i + w_S S_i$, where G_i is the trust of resource i, Rc_i, Ru_i, T_i, S_i are four trust attributes, and w_{Rc}, w_{Ru}, w_T, w_S are the weights for these attributes. Although the model in [7] is a multiple-attributes method, it use a manually approach to assign values to w_{Rc}, w_{Ru}, w_T, w_S (e.g., they were set as the same value 0.25). It is lack of adaptability in weight assignments for trust attributes.

Manuel et al. introduce a novel trust approach to evaluate the cloud resources by means of resource broker [8]. The resource broker chooses appropriate grid/cloud resource in heterogeneous environment based on the requirements of user. Their proposed trust management system is implemented with Kerberos authentication and PERMIS (PrivilEge and Role Management Infrastructure Standard) authorization to enhance the trust of the broker itself. Their trust enhanced resource broker evaluates the trust value of the resources based on the identity as well as behavioral trust. However, in [8], the authors only use two simple trust factors to quantify trustworthiness of resource, which may lead to incomplete or unfair outcome of trust decision-making. As one of the most dynamic and complex concepts in both social and business relationships, trust mechanism should involve many decision-making factors, such as real-time, dynamic, availability, reliability, security, and so on [34]–[36].

Fan and Perros propose a trust management framework for nulti-cloud environments [13], they address the problem of trust management in multi-cloud environments using a trust management architecture based on a group of distributed Trust Service Providers (TSPs). The proposed trust management framework for a multi-cloud environment is based on the proposed trust evaluation model and the trust propagation network. However, in [13], direct trust evaluation still use traditional subjective Probability method, rather than direct service behavior. How to adaptively fuse indirect trust and direct trust, this paper did not discuss in detail.

Ghosh et al. propose SelCSP: a framework to facilitate selection of cloud service providers [14]. It combines trustworthiness and competence to estimate risk of interaction. Trustworthiness is computed from personal experiences gained through direct interactions or from feedbacks related to reputations of vendors. Competence is assessed based on transparency in providers SLA guarantees. However, in SelCSP, trust evaluation still use traditional subjective ratings, rather than real-time service behavior.

Habib et al. propose a trust-aware framework to verify the security controls considering consumers' requirements [15]. The authors model the security controls in the form of trust properties. Then, they introduce a taxonomy of these properties based on their semantics and identify the authorities who can validate the properties. The taxonomy of these properties is the basis of trust formalisation in their proposed framework. Furthermore, a decision model is proposed as an integral part of the framework in order to empower consumers to determine trustworthiness of cloud providers. In [15], trust evaluation still is subjective method based on ratings.

Nagarajan and Varadharajan propose TESM: a trust enhanced secure model for trusted computing platforms [16]. The authors argue that given the nature of both binary and property based attestation mechanisms, an attestation requester cannot be absolutely certain if an attesting platform will behave as it is expected to behave. TESM uses a hybrid trust model based on subjective logic to combine 'hard' trust from measurements and properties and 'soft' trust from past experiences and recommendations to reduce such uncertainties. However, how to fuse these trust factors, e.g., hard trust or soft trust, this paper also did not discuss in detail.

Noor and Sheng propose the "Trust as a Service" (TaaS) framework to improve ways on trust management in cloud environments [17]. In particular, the authors introduce an adaptive credibility model that distinguishes between credible trust feedbacks and malicious feedbacks by considering cloud service consumers' capability and majority consensus of their feedbacks. However, this framework does not allow to assess trustworthiness based on monitoring information as well as users' feedback.

In the author's previous research [10], based on technology of distributed agents, trusted cloud service architecture is suggested for efficient scheduling cloud resources satisfying various user requests. The cloud service architecture aims to monitor servers dynamically and allocate high quality computing resources to users. The trusted data acquisition mechanism in this paper uses the monitored information of nodes in the cloud environment. This information consists of each node's spec information, resources usage, and response time. Then the model analyzes this information and prepares suitable resources on each occasion, and then allocates them immediately when user requests. As a result, cloud system can provide the high trustworthiness resources and high-level services based on the analyzed information and it is possible to utilize resources efficiently. In a recent work [11], the authors proposes a service operator-aware trust scheme (SOTS) for resource matchmaking across multiple clouds. However, both [10] and [11] only consider direct monitoring information without the user feedback information.

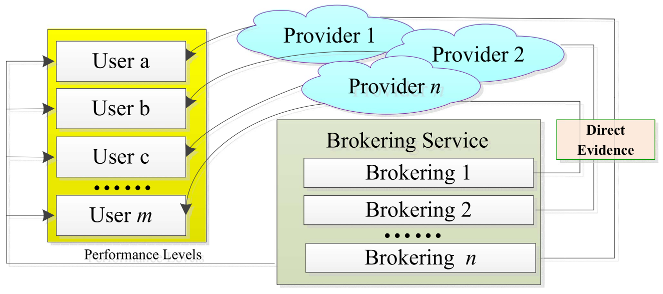

Fig. 1. Some existing brokering scenario without user feedback.

III. T-*Broker*'s Architecture

As mentioned in Part A of Section I, most current cloud brokering systems do not provide trust management capabilities to make trust decisions, which will greatly hinder the development of cloud computing. Fig.1 depicts the brokering scenario in existing brokers (e.g., RESERVOIR [18], PCMONS [19], RightScale [20], SpotCloud [21], and Aeolus [22]). We can see that this existing brokering architecture for cloud computing do not consider user feedback only relying on some direct monitoring information.

As depicted in Fig. 2, T-*broker* architecture, a service brokering system is proposed based on direct monitoring information and indirect feedbacks for the multiple cloud environment, in which T-*broker* is designed as the TTP for cloud trust management and resource matching. Before introducing the principles for assessing, representing and computing trust, we first present the basic architecture of T-*broker* and a brief description of its internal components.

A. Sensor-Based Service Monitoring (SSM)

This module is used to monitor the real-time service data of allocated resources in order to guarantee the SLA (Service Level Agreement) with the users. In the interactive process, this module dynamically monitors the service parameters and is responsible for getting run-time service data. The monitored data is stored in the evidence base, which is maintained by the broker. To calculating QoS-based trustworthiness of a resource [7], [10], we mainly focus on five

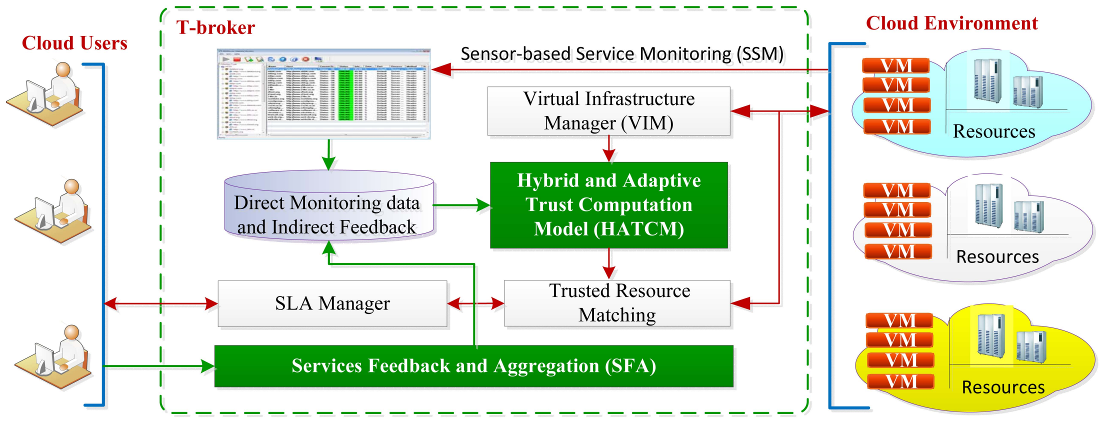
Cloud Users
T-broker
Sensor-based Service Monitoring (SSM)
Virtual Infrastructure Manager (VIM)
Direct Monitoring data and Indirect Feedback
Hybrid and Adaptive Trust Computation Model (HATCM)
SLA Manager
Trusted Resource Matching
Services Feedback and Aggregation (SFA)
Cloud Environment
VM
Resources

Fig. 2. T-*broker*'s Architecture and main function modules.

kinds of trusted attributes of cloud services, which consists of node spec profile, average resource usage information, average response time, average task success ratio, and the number of malicious access. The node spec profile includes four trusted evidences: CPU frequency, memory size, hard disk capacity and network bandwidth. The average resource usage information consists of the current CPU utilization rate, current memory utilization rate, current hard disk utilization rate and current bandwidth utilization rate. The number of malicious access includes the number of illegal connections and the times of scanning sensitive ports.

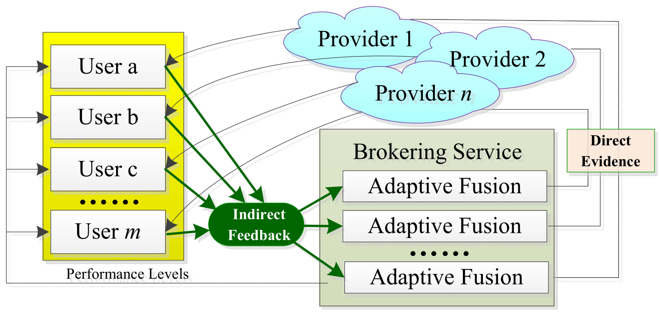

Fig. 3. T-*broker*' brokering scenario with adaptive fusion mechanism.

B. Virtual Infrastructure Manager (VIM)

Each cloud provider offers several VM configurations, often referred to as instance types. An instance type is defined in terms of hardware metrics such as CPU frequency, memory size, hard disk capacity, etc. In this work, the VIM component is based on the OpenNebula virtual infrastructure manager [42], [43], this module is used to collect and index all these resources information from multiple cloud providers. It obtains the information from each particular cloud provider and acts as a resource management interface for monitoring system. Cloud providers register their resource information through the VIM module to be able to act as sellers in a multi-cloud marketplace. This component is also responsible for the deployment of each VM in the selected cloud as specified by the VM template, as well as for the management of the VM life-cycle. The VIM caters for user interaction with the virtual infrastructure by making the respective IP addresses of the infrastructure components available to the user once it has deployed all VMs.

C. SLA Manager and Trusted Resource Matching

In the multiple cloud computing environment, SLA can offer an appropriate guarantee for the service of quality of resource providers, and it serves as the foundation for the expected level of service between the users and the providers [45], [46]. An SLA is a contract agreed between a user and a provider which defines a series of service quality characters. Adding trust mechanism into the SLA management, cloud brokering system can prepare the best trustworthiness resources for each service request in advance, and allocate the best resources to users. In general, the service resource register its services on the cloud brokering system. The service user negotiates with the service provider about the SLA details; they finally make a SLA contract. According to the SLA contract, the resource matching module selects and composites highly trusted resources to users from the trusted resource pool.

D. Hybrid and Adaptive Trust Computation Model (HATCM)

Trust and feedback management systems are successfully used in numerous application scenarios to support users in identifying the reliable and trustworthy providers. Similar approaches are needed to support cloud brokering systems in matching appropriate trustworthy resources from different providers in a multi-cloud computing environment. HATCM module is not only the core of the trust-aware cloud computing system, but also a key task of this work. Using this module, the middleware architecture can sort high performance resources through analyzing the history information of the resources for providing highly trusted resources dynamically.

As depicted in Fig. 3, HATCM uses a hybrid and adaptive trust model to compute the overall trust degree of service resources, in which trust is defined as a fusion evaluation result from adaptively combining real-time service behavior with the social feedback of the service resources. The HATCM allows

TABLE I
NOTATIONS AND THEIR MEANINGS

Notations	Description
$\Delta\lambda$	the time window for trust evaluation
t	the time stamp for direct evidence gathering, $t \in [1, \Delta\lambda]$
d_{tm}	a gathered service evidence at t, the value of m-th QoS indicators at t
$\mathbf{d}_t$	a vector composed by the collecting evidence, $\mathbf{d}_t = \{d_{t1}, d_{t2}, \cdots, d_{tm}\}$
$D(\Delta\lambda)$	the set of measurement data at time window $\Delta\lambda$
e'_{tk}	the normalization value of d_{tm} according to vector normalization method
e''_{tk}	the normalization value of d_{tm} according to linear transformation method
$E_\delta(\Delta\lambda)$	the standardized decision matrix based on vector normalization method
$E_\phi(\Delta\lambda)$	the standardized decision matrix based on linear transformation method
$\mathbf{w}$	the weight vector, $\mathbf{w} = \{w_1, w_2, \cdots, w_m\}$
$\mathbf{u}_t$	a vector composed by fuzzy adjustment coefficient.
R	the set of trusted service resources, $R = \{r_1, r_2, \cdots, r_p\}$
r_i	the i-th trusted service resource in R.
$\alpha_t(r_i)$	the instant trust degree of r_i at time t
$\mathbb{R}_{\Delta\lambda}(\alpha_t)$	the real-time trust degree
$\mathbb{F}_{\Delta\lambda}(r_i)$	the feedback trust degree
$\mathbb{O}_{\Delta\lambda}(r_i)$	the overall trust degree (OTD)
γ	the weight factor for $\mathbb{R}_{\Delta\lambda}(\alpha_t)$ and $\mathbb{F}_{\Delta\lambda}(r_i)$
$\varepsilon_{r_i}(\Delta\lambda)$	the mean absolute deviation (MAD)
$\xi_{r_i}(\Delta\lambda)$	the mean absolute percent error (MAPE)

cloud users to specify their requirements and opinions when accessing the trust score of cloud providers. That is, users can specify their own preferences, according to their business policy and requirements, to get a customized trust value of the cloud providers.

E. Services Feedback and Aggregation (SFA)

In large-scale distributed systems, such as grid computing, P2P computing, wireless sensor networks, and so on, feedback provides an efficient and effective way to build a social-evaluation-based trust relationship among network entities. By the same token, feedback also can provider important reference in evaluating cloud resource trustworthiness. Consider large-scale cloud collaborative computing environment which host hundreds of machines and handles thousands of request per second, the delay induced by trust system can be one big problem. So, there is no doubt that the computational efficiency of a feedback aggregating mechanism is the most fundamental requirement. As depicted in Fig. 3, we build cloud social evaluation system using feedback technology among virtualized data centers and distributed cloud users, and we use a lightweight feedback mechanism, which can effectively reduce networking risk and improve system efficiency.

IV. CLOUD TRUST COMPUTATION MODEL

This section contains a lot of notations. At the beginning of this section, we list some key notations and their meanings in Table I to make it easier for the readers to follow up.

TABLE II
QoS INDICATORS (OR SERVICE BEHAVIOR)

Trust attributes	QoS indicators (service behavior)
node spec profiles	CPU frequency memory size hard disk capacity network bandwidth
average resource usage information	current CPU utilization rate current memory utilization rate current hard disk utilization rate current bandwidth utilization rate
average response time	average response time
average task success ratio	average task success ratio
the number of malicious access	the number of illegal connections the times of scanning sensitive ports

Trustworthiness computation model and approaches are the core technologies of trust management. In our T-*broker*, we propose a hybrid and adaptive trust model to compute the overall trust degree of service resources, in which trust is a fusion evaluation result from adaptively combining real-time trust computation with the feedback of the service resources. In this section, we first present the adaptive real-time trust computation approach based on multiple key trust attributes of service resources. We then introduce the light-weight feedback trust computation approach. Finally, we discuss how to aggregate the two kind of trust factors with an adaptive mechanism.

A. Adaptive Real-Time Trust Computation

As mentioned in Part A of section III, to calculating resource trust degree from the perspective of QoS guaranteeing, we mainly focus on five kinds of trusted attributes of cloud services [7], [10], [11], which consists of node spec profile, average resource usage information, average response time, average task success ratio and the number of malicious access. Both the node spec profile and average resource usage information include four trusted evidences. Thus, there are total 12 QoS indicators (or service behavior) needed to be acquired, quantified and analyzed, which is summarized in Table II.

Monitoring service behavior is the process of acquiring state information from a cloud resource. In traditional poll-based approach, monitoring is performed by a management node, which periodically polls nodes in its domain for the values of related parameters. An alternative to the poll-based approach, is the push approach, whereby network nodes send values of parameters to the manager whenever changes to those values occur. With the emergence of large and dynamic networked systems, the push approach is gaining importance, because this approach enables the management system to continuously follow the evolution of the network state [44]. In this work, we adopted the push-based approach to acquire these QoS indicators. We deployed two types of software sensors: (i) Monitoring sensors are responsible for collecting the direct performance indicators of computing resources. Such as CPU frequency, memory size, hard disk capacity, the number of illegal connections, the times of scanning sensitive ports etc. (ii) Computing sensors are responsible for collecting and computing the indirect QoS indicators, which need real-time calculation and statistics. Such as the current CPU utilization rate, current memory utilization rate, current hard disk utilization rate, current bandwidth utilization rate, average response time and average task success ratio.

The historical value of trusted data obtained by software sensors can be arranged in a time window $\Delta\lambda$, and $\Delta\lambda$ is a set of time slot units. If a time window (such as one hour) is divided by 5 minutes, then $\Delta\lambda = \{1, 2, \ldots, t, \ldots, 12\}$. At a time window $\Delta\lambda$, supposed that we have obtained $\Delta\lambda$ groups of measurement data $D(\Delta\lambda) = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_t, \ldots, \mathbf{d}_{\Delta\lambda}\}$, where $1 \le t \le \Delta\lambda$ and $\mathbf{d}_t = \{d_{t1}, d_{t2}, \ldots, d_{tm}\}$. Thus, an order characteristic matrix is formed as:

$$D(\Delta\lambda) = \begin{pmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \\ \vdots \\ \mathbf{d}_{\Delta\lambda} \end{pmatrix} = \begin{pmatrix} d_{11} & d_{12} & \ldots & d_{1m} \\ d_{21} & d_{22} & \ldots & d_{2m} \\ & & \ddots & \\ d_{\Delta\lambda 1} & d_{\Delta\lambda 2} & \ldots & d_{\Delta\lambda m} \end{pmatrix} \tag{1}$$

Data normalization is a process in which data attributes within a data model are organized to increase the cohesion of entity types. In other words, the goal of data normalization is to reduce and even eliminate data redundancy, an important consideration for application developers because it is incredibly difficult to stores objects in a relational database that maintains the same information in several places.

Thus, before clustering analysis of trusted data, the data set should be normalized to eliminate deviation caused by each attribute item's difference of unit and value domain. In this paper, we calculating weights based on the maximizing deviation method [37]–[39]. Let $\mathbf{d}_k = \{d_{1k}, d_{2k}, \ldots, d_{tk} \ldots, d_{\Delta\lambda k}\}$ denote all column vector of trust indicator k, where $k = \{1, 2, \ldots, m\}$. Thus, another form of the matrix $D(\Delta\lambda)$ can be expressed as:

$$\begin{aligned} D(\Delta\lambda) &= \begin{pmatrix} \mathbf{d}_1 & \mathbf{d}_2 & \cdots & \mathbf{d}_k \end{pmatrix} \\ &= \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1m} \\ d_{21} & d_{22} & \cdots & d_{2m} \\ & & \ddots & \\ d_{\Delta\lambda 1} & d_{\Delta\lambda 2} & \cdots & d_{\Delta\lambda m} \end{pmatrix} \end{aligned} \tag{2}$$

For any column data $d_{tk} \in \mathbf{d}_k$ (where $\mathbf{d}_k$ is the k-*th* column vector in Eq. (2)), we use following two models to compute its normalized value:

(1) Vector normalization method:

$$e'_{tk} = \begin{cases} d_{tk} \Big/ \sqrt{\sum_{t=1}^{\Delta\lambda} d_{tk}^2}, & (a) \\ (1/d_{tk}) \Big/ \sqrt{\sum_{t=1}^{\Delta\lambda} (1/d_{tk})^2}, & (b) \end{cases} \tag{3}$$

where condition (a) means that d_{tk} is a positive increasing value, i.e., the large value of d_{tk} is what we expect; this value covers CPU frequency, memory size, hard disk capacity, network bandwidth, average task success ratio, etc. Condition (b) means that d_{tk} is a positive decreasing value, i.e., the small value of d_{tk} is what we expect; this value covers average response time. Through Eq. (3), we can get a standardized decision matrix $E_\delta(\Delta\lambda)$:

$$E_\delta(\Delta\lambda) = \begin{pmatrix} \mathbf{e}'_1 \\ \mathbf{e}'_2 \\ \vdots \\ \mathbf{e}'_{\Delta\lambda} \end{pmatrix} = \begin{pmatrix} e'_{11} & e'_{12} & \cdots & e'_{1m} \\ e'_{21} & e'_{22} & \cdots & e'_{2m} \\ & & \ddots & \\ e'_{\Delta\lambda 1} & e'_{\Delta\lambda 2} & \cdots & e'_{\Delta\lambda m} \end{pmatrix} \tag{4}$$

(2) Linear transformation method:

$$e''_{tk} = \begin{cases} d_{tk}/max(d_{tk}), & (a) \\ d_{tk}/min(d_{tk}), & (b) \end{cases} \tag{5}$$

where $max(d_{tk})$ and $min(d_{tk})$ are, respectively, the maximum and minimum value of all row data d_{tk}. condition (a) means that d_{tk} is a positive increasing value, and condition (b) means that d_{tk} is a is a positive decreasing value. Through Eq. (5), we can get another standardized decision matrix $E_\phi(\Delta\lambda)$:

$$E_\phi(\Delta\lambda) = \begin{pmatrix} \mathbf{e}''_1 \\ \mathbf{e}''_2 \\ \vdots \\ \mathbf{e}''_{\Delta\lambda} \end{pmatrix} = \begin{pmatrix} e''_{11} & e''_{12} & \cdots & e''_{1m} \\ e''_{21} & e''_{22} & \cdots & e''_{2m} \\ & & \ddots & \\ e''_{\Delta\lambda 1} & e''_{\Delta\lambda 2} & \cdots & e''_{\Delta\lambda m} \end{pmatrix} \tag{6}$$

Through normalization of trusted data, all the monitored data are expressed within [0, 1]. We can get a new matrix $E(\Delta\lambda) = (e_{tk})_{\Delta\lambda \times m}$, and $E(\Delta\lambda)$ is called a standardized decision matrix. We use Eqs.(3) and (5) to compute standardized matrix respectively, then we can get two standardized decision matrix $E_\delta(\Delta\lambda) = (e'_{tk})_{m\times\Delta\lambda}$ and $E_\phi(\Delta\lambda) = (e''_{tk})_{m\times\Delta\lambda}$.

Using standardized decision matrix $E_\delta(\Delta\lambda)$, we can get decision results $\delta_t(\mathbf{w}) = \sum_{k=1}^{m} (w_k e'_{tk})^2$, and based on standardized decision matrix $E_\phi(\Delta\lambda)$, we can get decision results $\phi_t(\mathbf{w}) = \sum_{k=1}^{m} (w_k e''_{tk})^2$, where $\mathbf{w} = \{w_1, w_2, \ldots, w_m\}$ is the weight vector. Then, we can use the following equation to reflect the integrated decision results:

$$L(\mathbf{u}_t, \mathbf{w}) = u_t^2 \delta_t(\mathbf{w}) + (1 - u_t)^2 \phi_t(\mathbf{w}) \tag{7}$$

where $\mathbf{u}_t$ is fuzzy adjustment coefficient. According to maximizing deviation method for multi-indices decisions [38], [39], if there is a large difference between decision results $\delta_t(\mathbf{w})$ and $\phi_t(\mathbf{w})$, such decision results (attributes) will play an important role and should be given greater weight coefficients. Thus, weight calculation problem can be converted to an optimal programming model [38], [39]:

$$\begin{aligned} &max(F(\mathbf{u}_t, \mathbf{w}) \\ &\quad = \sum_{t=1}^{\Delta\lambda} L(\mathbf{u}_t, \mathbf{w})), \quad \begin{cases} 0 \le u_t \le 1, \; t \in [1, \Delta\lambda] \\ \sum_{k=1}^{m} w_k = 1, \; k \in [1, m] \\ w_k \ge 0 \end{cases} \end{aligned} \tag{8}$$

To reduce the number of unknown equation and simplify calculation process, constraint conditions u_t, $t \in [1, n]$ and w_k, $k \in [1, m]$ can be not considered at first computing stage. Then, the Lagrangian function is constructed as:

$$L(\mathbf{u}, \lambda) = \sum_{t=1}^{n} L(\mathbf{u}_t, \mathbf{w}) + \lambda(\sum_{k=1}^{n} w_k - 1) \tag{9}$$

Computing the partial derivative of $L(\mathbf{u}, \lambda)$ according to u_t, w_k and λ respectively. And set them to zero, we can obtain:

$$\begin{cases} \dfrac{\partial L(\mathbf{u}, \lambda)}{\partial w_k} = 2w_k \sum_{t=1}^{\Delta\lambda} (u_t^2 (e'_{tk})^2 + (1-u_t^2)(e''_{tk})^2 + \lambda) = 0 \\ \dfrac{\partial L(\mathbf{u}, \lambda)}{\partial u_t} = 2u_t \sum_{k=1}^{m} (w_j^2 (e'_{tk})^2 + (1-u_t)(e''_{tk})^2) = 0 \\ \dfrac{\partial L(\mathbf{u}, \lambda)}{\partial \lambda} = \sum_{k=1}^{m} w_k - 1 = 0 \end{cases} \quad (10)$$

Computing Eq. (10), we can get:

$$w_k = \left(\sum_{i=1}^{m} \left(\frac{\sum_{t=1}^{\Delta\lambda} (u_t^2 (e'_{tk})^2 + (1-u_t^2)(e''_{tk})^2}{\sum_{t=1}^{\Delta\lambda} (u_t^2 (e'_{ti})^2 + (1-u_t^2)(e''_{ti})^2} \right)\right)^{-1} \quad (11)$$

based on above maximizing deviation model, through several iterations, we can get weight vector $\mathbf{w} = \{w_1, w_2, \ldots, w_m\}$.

Let $R = S \cup O = \{r_1, r_2, \ldots, r_p\}$ denotes p trusted service resources in the multi-cloud system (we can define a threshold of trust in the process of resource matching), where S is the set of available resources and O is the resources in line with user expectations, and R is called trusted resource domain, S is called service resource domain and O is called user requirement domain. In a multi-cloud system, $\exists r_i \in R$, let $\alpha_t(r_i)$ denote the instant trust degree of r_i at time-stamp t. In this work, we define $\alpha_t(r_i)$ as:

$$\alpha_t(r_i) = \mathbf{e}_t \cdot \mathbf{w} = \sum_{k=1}^{m} e_{tk} w_k \quad (12)$$

where $\mathbf{e}_t = \{e_{tk}\} = \{(e'_{tk} + e''_{tk})/2\}$, $e_{tm} \in [0, 1]$ (subscript t is the time point or time stamp of this measurement). The vector $\mathbf{e}_t$ is called an evidence of trust evaluation, and it has m-dimensional measuring indicators $d_{t1}, d_{t2}, \ldots, d_{tm}$. $\mathbf{w}$ is the weight vector assigned to these trusted indicators.

In the past $\Delta\lambda$ time slots (or in the time window $\Delta\lambda$), using Eq.(12), we can get a time series $\overline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_t, \ldots, \alpha_{\Delta\lambda}\}$. Real-time trust degree (RTD) can be calculated by following formula:

$$\mathbb{R}_{\Delta\lambda}(\alpha_t) = \overline{\alpha} \times \mathbf{s} = \sum_{t=1}^{\Delta\lambda} (\alpha_t(r_i) \times s_t(r_i)) \quad (13)$$

where $\mathbf{s} = \{s_t(r_i)\} = \{s_1(r_i), s_2(r_i), \ldots, s_{\Delta\lambda}(r_i)\}$, and $\sum_{t=1}^{\Delta\lambda} s_t(r_i) = 1$. $s_t(r_i) \in [0, 1]$ is the weights assigned to each instant trust degree $\alpha_t(r_i)$. According to human social behavior habits [31], old knowledge has less infection, whereas new knowledge has more contribution to trust decision-making. Thus we can define the $\mathbf{s} = \{s_t(r_i)\} = \{s_1(r_i), s_2(r_i), \ldots, s_{\Delta\lambda}(r_i)\}$ as an attenuation function:

$$s_t(r_i) = \frac{(1 - (t+1)^{-1})}{\sum_{t=1}^{\Delta\lambda} (1 - (t+1)^{-1})} \quad (14)$$

where $t \in [1, \Delta\lambda]$. According to [31], trustworthiness should meet a fundamental property: the time-based attenuation. In this study, the computing expression for RTD should reflect this fundamental property. At the same time, $s_t(r_i)$ should meet normalization. Following, we use Theorems 1 and 2 to prove the two important properties.

Theorem 1: Through Eq. (14), we can get a weight vector $\{s_t(r_i)\} = \{s_1(r_i), s_2(r_i), \ldots, s_{\Delta\lambda}(r_i)\}$. *Supposed that there are two time-stamps t_x and t_y. If ($t_x < t_y$), we can get $s_{t_x}(r_i) < s_{t_y}(r_i)$, which means that Eq. (14) meets the time-based attenuation property.*

Proof: see **Appendix A**.

Theorem 2: The weight vector $\{s_t(r_i)\}$ calculated by Eq. (14) meets the normalization, that is, for any $t \in [1, \Delta\lambda]$, $\sum_{t=1}^{\Delta\lambda} s_t(r_i) = 1$ and $s_t(r_i) \in [0, 1]$.

Proof: see **Appendix B**.

B. Light-Weight Feedback Trust Computation

The feedback system collects locally-generated users' ratings and aggregates these ratings to yield the global evaluation scores. After a user completes a transaction, the user will provide his or her rating as a reference for other users in future transactions.

In [40], Whitby and Jøang proposed the beta feedback system which is based on the theory of statistics and is characterized by flexibility and simplicity. Inspired by [40], we use the beta probability density functions to compute feedback trust degree:

$$\mathbb{F}_{\Delta\lambda}(r_i) = \frac{\zeta + 1}{\zeta + \xi + 2} \quad (15)$$

where ζ is the number of positive ratings (>0.5) towards service r_i, and ξ is the number of negative ratings (<0.5) towards service r_i. In Eq. (15), if ($\zeta + \xi = 0$), this means that r_i is a new joined cloud provider. Referring to [41], we set $\mathbb{F}_{\Delta\lambda}(r_i) = 0.5$ when ($\zeta + \xi = 0$). The idea is based on a research result in [41], in which the authors pointed out that suspicion of new provider is socially inefficient since only a limited number of providers are malicious in the cloud market. This approach can give a chance for new cloud providers to enter the cloud market until they are proved untrustworthy.

As mentioned in Part E of section III, the brokering system need to collect huge amount of users' feedbacks and aggregates them to yield the global evaluation for the providers. At the same time, multiple cloud system host hundreds of machines and handles thousands of request per second, the delay induced by trust system can be one big problem. From Eq. (15), we can see that our feedback trust computation is a lightweight approach, in which only simple arithmetic operations and counting operation are involved.

C. Hybrid and Adaptive OTD Aggregation

We adopt the idea that overall trust degree (OTD) comprises two parts: First-hand trust (trust based on real-time and multi-source service data) and second-hand trust (feedback) [35], [36]. This hybrid trust calculation approach is based on a combination of two kinds of known trust methodologies: feedback-based trust and experience-based trust. The key idea is that by combining two different methodologies, the resulting integrated framework improves some weaknesses of the constituent methodologies and, consequently, the overall

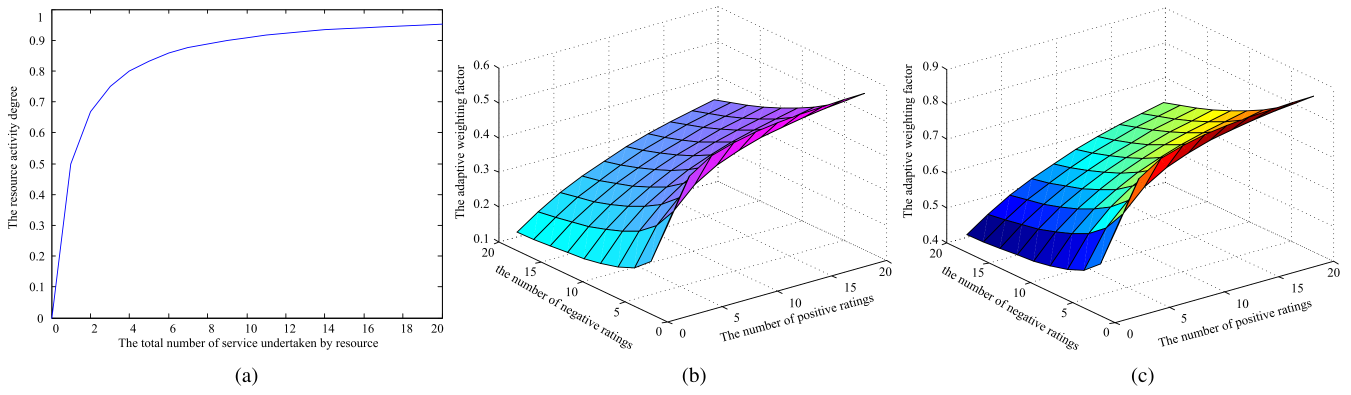

Fig. 4. The resource activity degree $\rho(r_i)$ and the weight factor γ. (a) The values of $\rho(r_i)$ with changes of $L(r_i)$. (b) The values of γ, where $\rho(r_i) = 0.2$. (c) The values of γ, where $\rho(r_i) = 0.8$.

assessment of cloud services. In the proposed trust scheme, the resource r_i's OTD is calculated as follows:

$$\mathbb{O}_{\Delta\lambda}(r_i) = (1-\gamma) \times \mathbb{R}_{\Delta\lambda}(r_i) + \gamma \times \mathbb{F}_{\Delta\lambda}(r_i) \tag{16}$$

where $(1-\gamma)$ is the weight of $\mathbb{R}_{\Delta\lambda}(r_i)$, correspondingly, γ is the weight of $\mathbb{F}_{\Delta\lambda}(r_i)$. If we set $\gamma = 1$, the weight of $\mathbb{R}_{\Delta\lambda}(r_i)$ is 0, and the proposed model will degenerate to the traditional feedback system. However, many studies show that real-time trust $\mathbb{R}_{\Delta\lambda}(r_i)$ is a helpful component in building a dependable trust relationship [7], [9], [10]. When the system is highly dynamic and most users are bad, this value of $\mathbb{R}_{\Delta\lambda}(r_i)$ should be set with a high weight automatically.

Now, the key question is, how much value of γ is reasonable and accurate? In the vast majority of the current works [26]–[28], γ is allocated through three subjective ways: experts opinion method, random allocation method and average weight method. However, all of the three methods are subjective ways, which has two key deficiencies: these methods does not reflect trust decision scientific and reasonable, and may lead to misjudgment of trust decision; these methods are lack of adaptability, once the value of γ is identified, it will be difficult to adjust by system dynamically in a practical distributed application environment. In the following section, we will introduce two new adaptive weight allocation functions, confidence factor and feedback factor, which can solve the difficulty of the weight allocation.

Intuitively, the value of $\mathbb{F}_{\Delta\lambda}(r_i)$ calculated above should have a higher weight if the number of rating users is higher. Likewise users with more interactions with a particular service should have a higher say in recommendation. Now we use a new function to reflect these objective phenomenon. The weight factor γ is defined as:

$$\gamma = \frac{\rho(r_i) + q(r_i)}{2} = \frac{\rho(r_i) + \frac{\zeta}{\zeta+\xi}}{2} \tag{17}$$

where the function $q(r_i) = \frac{\zeta}{\zeta+\xi}$, which can reflect the proportion of positive feedback in all feedback from users. ζ is the number of positive ratings (> 0.5) towards service r_i, and ξ is the number of negative ratings (<0.5) towards service r_i. The expression $\rho(r_i)$ is called activity degree factor of the resource r_i which is defined as follow:

$$\rho(r_i) = 1 - \frac{1}{\mathbb{L}(r_i) + 1} \tag{18}$$

where $\mathbb{L}(r_i)$ is the total number of service undertaken by resource r_i. The function $\rho(r_i)$ has the desirable property that with increasing $\mathbb{L}(r_i)$ ($\mathbb{L}(r_i)$ is a positive integer), the function quickly approaches 1 (see Fig. 4(a)). Substituting Eq. (18) into Eq. (17), we can get:

$$\gamma = \frac{\rho(r_i)(\zeta+\xi)+\zeta}{2(\zeta+\xi)} = \frac{\mathbb{L}(r_i)(\zeta+\xi)+\zeta(\mathbb{L}(r_i)+1)}{2(\zeta+\xi)(\mathbb{L}(r_i)+1))} \tag{19}$$

From Eq. (17), γ can get a larger value if the value of $q(r_i)$ closes to 1. We use sub-figures (b) and (c) in Fig. 4 to show the trends of weight factor γ with different values of $\rho(r_i)$, ζ and ξ. The results in Fig. 4 show that γ's values are proportional to $\rho(r_i)$ and ζ, and inversely proportional to ξ, which clearly reflects the design goal of this work.

From Eqs. (17), (18) and (19), the computation mechanisms for γ show that the resource r_i is more active and reliable in the network environment if the value of $\mathbb{L}(r_i)$ or ζ is larger. Following, we use Theorems 3 and 4 to prove this important virtue brought by Eqs (17), (18) and (19).

Theorem 3: the weight factor γ can get a larger value if the value of $\rho(r_i)$ closes to 1.

Proof: see **Appendix C**.

Theorem 4: the weight factor γ can get a larger value if the value of $q(r_i)$ closes to 1.

Proof: see **Appendix D**.

V. Experiments and Evaluation

In this section, we first describe how to set up the experimental prototype system in an Eucalyptus-based cloud environment, Then, the experimental results are described.

A. Experimental Methods

To evaluate the trust scheme based on technologies introduced in Section IV, we set up an experimental prototype system based on Eucalyptus cloud environment [47]. Performance is examined from two aspects: (1) Accuracy, which is used to check whether the proposed model and its related algorithms can accurately and consistently provide trust measurement [30]; (2) Capacity in dealing with dynamic behavior, which is used examines our model's ability to track the dynamic behavior of cloud services.

A universal trust system should have good accuracy for trust measurement, that is, it should be able to detect unqualified resources. In this subsection, we first introduce two formulaes for calculating deviation to reflect the system's accuracy, which is the difference between the actual quantity and the calculating deviation. The two mechanisms are shown as follows [30], [48]:

For a given cloud resource (service) r_i, its mean absolute deviation (MAD) is defined as:

$$\varepsilon_{r_i}(\Delta\lambda) = \frac{\sum_{t=1}^{\Delta\lambda} y_t}{\Delta\lambda} \tag{20}$$

where y_t is the calculating deviation at time t, $y_t = (\mathbb{O}_{t+1} - \mathbb{O}_t)$. $\mathbb{O}_{t+1}$ is the actual value calculated by our trust system at time $(t+1)$, and $\mathbb{O}_t$ is the calculated value by our trust system at time t. $\Delta\lambda$ is the time window, which equals total number of periods of evaluation.

Although $\varepsilon_{r_i}(\Delta\lambda)$ can reflect the evaluation accuracy, it is difficult to reflect the unbiasedness in the measurement. We can use mean absolute percent error (MAPE) to reflect the unbiasedness of a forecast result. For a given cloud resource (service) N_i, its MAPE is defined as:

$$\xi_{r_i}(n) = \frac{1}{\Delta\lambda}\sum_{t=1}^{\Delta\lambda}\frac{y_t}{\mathbb{O}_{t+1}}(\times 100\%) \tag{21}$$

As mentioned in Eq. (20), y_t is the calculating deviation at time t, $y_t = (\mathbb{O}_{t+1} - \mathbb{O}_t)$. $\mathbb{O}_{t+1}$ is the actual value calculated by our trust system at time $(t+1)$, and $\mathbb{O}_t$ is the value calculated by our trust system at time t. $\Delta\lambda$ is the total number of periods of evaluation. The value of $\varepsilon_{r_i}(\Delta\lambda)$ and $\xi_{r_i}(\Delta\lambda)$ are indicators of bias in the calculations. They are checked to determine if they are within the acceptable control limits. In the experiments, each resource's log information is recorded at the same time interval (60 seconds). We collect about 1,000 groups of actual status log data as the training samples (100 groups of data for each machine).

Although there are many trust mechanisms (see [14]–[16]) proposed for cloud computing environment, they use traditional subjective method rather than real-time service behavior (as presented in Section II). Thus, these trust evaluation models are inconsistent with T-*broker* (in this paper, we adopt the behavior-based trust computing method). In one experimental environment, to compare the trust models using different mechanisms is a challenging and difficult task. In order to compare proposed trust system with existing trust mechanisms, we select two existing behavior-based trust models, which is presented as follows:

(a) The subjective trust model (STM), which is similar to Kim's model [7], uses an average approach to assign weights to trust attributes, so we call it a subjective trust model. Following is this trust model:

$$\mathbb{O}_{r_i} = \frac{\sum_{k=1}^{m} \mathbb{I}(k)}{m} \tag{22}$$

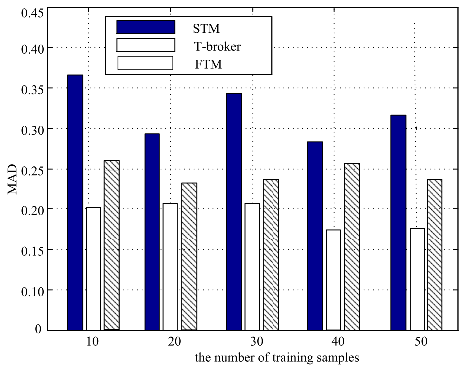

Fig. 5. The values of MAD with a small number of training samples.

where $\mathbb{O}_{r_i}$ is the average-based trust value of the resource r_i, $\mathbb{I}(k), k \in [1, m]$ is the k^{th} indicator for trust measurement, and m is the total number of these trust indicators. Obviously, STM is a subjective measurement approach, it assigns equal weights to each trust indicators.

(b) The fuzzy trust model (FTM), such as Stefan's model [26] and Wu's model [29], uses fuzzy-logic to assign weights to trust attributes. Following is Wu's trust model:

$$\mathbb{O}_{r_i} = \{w_1, w_2, \ldots, w_m\} \circ \{u_1, u_2, \ldots, u_m\} \tag{23}$$

where $\circ$ is the fuzzy compose operator, $\{u_1, u_2, \ldots, u_m\}$ is the indicator vector and $\{w_1, w_2, \ldots, w_m\}$ is the weight vector. In the fuzzy trust model, its trust calculation expression is somewhat similar to ours; however, the weight vector $\{w_1, w_2, \ldots, w_m\}$ is computed by fuzzy-logic theory.

B. Accuracy Evaluation

As described in Eq. (20), the value of $\varepsilon_{r_i}(\Delta\lambda)$ is used to measure the degree of deviation of calculating results; thus, the closer its value is to zero, the higher the calculating accuracy. First, observing MAD under conditions with different number of training samples (Note: we gather a training sample $d_t = (d_{t1}, d_{t2}, \ldots, d_{tm})$ at each time-stamp t, so the number of training samples equals to the number of time-stamps). In order to observe experimental results under different scale of training samples, we use two kinds of inputting samples, a small number of training samples and a large number of training samples.

In the first group of experiments, the total number of training samples changes from 10 to 50. Fig. 5 shows that the number of training samples has a direct effect on the accuracy of the trust models. When the number of training samples is small ($t < 30$), the MADs of three models are more than 0.20. When the number of training samples is set larger ($t \geq 30$), the MADs of the other two models are more than 0.23. The MAD of our trust model is less than 0.20. From the results in Fig. 5, the MAD of our trust model is much smaller than that of STM and FTM, which reflects that our model's performance is better than that of other models under conditions with different number of training samples. As an example, see the

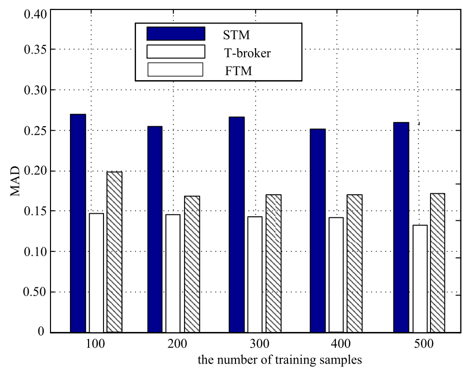

Fig. 6. The values of MAD with a large number of training samples.

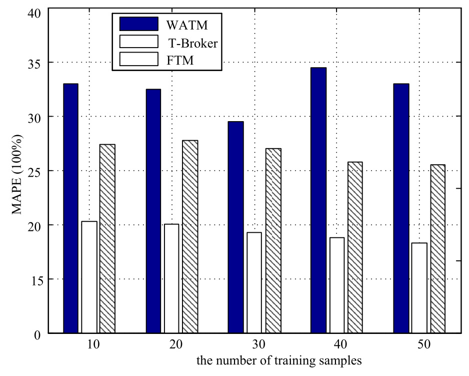

Fig. 7. The values of MAPE with a small number of training samples.

results in Fig. 5, when number of training samples is 40, the MAD of our trust model is 0.1781, the MAD of STM is 0.2815 and the MAD of FTM is 0.2617. Obviously, our model's accuracy is higher than FTM nearly 9.07 percentage points, and is higher than STM nearly 10.34 percentage points.

Fig. 6 shows the second group of experimental results with the total number of training samples changing from 100 to 500. From the results in Fig. 6, the MAD of our T-*broker* model is much smaller than that of STM and FTM, which reflects that our model's performance is also better than that of other models under conditions with a large number of training samples. As an example, see the results in Fig. 6, when number of training samples is 200, the MAD of T-*broker* is 0.1462, the value of STM is 0.2601 and the value of FTM is 0.1721.

The MAPE is a measure of accuracy in a fitted time series value in statistics, specifically trending. It usually expresses accuracy as a percentage. MAPE can reflects the unbiasedness of the calculating model. A smaller value of MAPE reflects the calculating model has better and unbiased accuracy. We also use two kinds of inputting samples to evaluate the MAPE of the three models, a small number of training samples and a large number of training samples.

Fig. 7 shows the MAPE's values of the three trust mechanisms under different number of training samples. In the first group of experiments, the total number of training samples is set as the same as Fig. 5, which increases from 10 to 50. From the results in Fig. 7, the MAPE of our T-*broker* model is much smaller than that of STM and FTM,

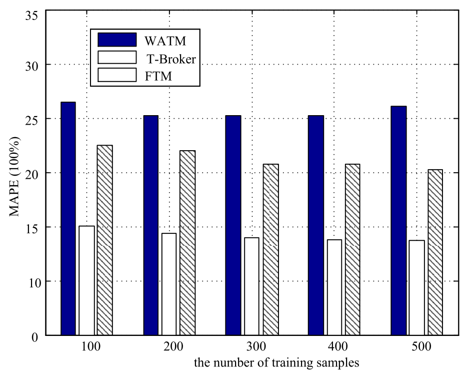

Fig. 8. The values of MAPE with a large number of training samples.

which reflects that our model's performance is also better than that of other models under conditions with a small number of training samples. As an example, see the results in Fig. 7, when number of training samples is 30, the MAPE of T-*broker* is 19.12, the value of STM is 29.10 and the MAD of FTM is 27.01.

Fig. 8 shows the MAPE's values of the three trust mechanisms under a large number of training samples. In this group of experiments, the total number of training samples is set as the same as Fig. 6, which increases from 100 to 500. From the results in Fig. 8, the MAPE of our T-*broker* is also much smaller than that of STM and FTM, which reflects that our model's performance is better than that of other models under conditions with a large number of training samples.

Further analysis of results from Fig. 5 to Fig. 8, we can find that, under different scale of training samples, the trends of our trust scheme about both MAD and MAPE are slightly smooth and the trends of FMT and STM are slightly steep, which indicates that the number of training samples has less effect on the performance of our model. From the view of computational efficiency, using fewer training samples, our trust scheme can achieve a better evaluation accuracy, which is an important advantage of this trust scheme. Under an actual multiple cloud computing environment, we suggest that the number of training samples should not be too large so that the trust system can save running time and significantly improve the system's rapid response capability.

C. Dealing With Dynamic Service Behavior

A cloud resource can either be a good or bad service provider. The dynamism of a cloud service means that a service resource can dynamically change its behavior as providing good or bad service. QoS-aware trust indicators can be divided into two types: static indicators and dynamic indicators. The current CPU utilization rate, current memory utilization rate, current hard disk utilization rate, current bandwidth utilization rate, average response time and average task success ratio are dynamic indicators; the CPU frequency, memory size, and hard disk capacity are static indicators. To model the dynamic indicators of a service provider, we introduce two dynamic behavior profiles:

1) The random walk profile is used to model the generally more or less predictable behavior of a cloud service.

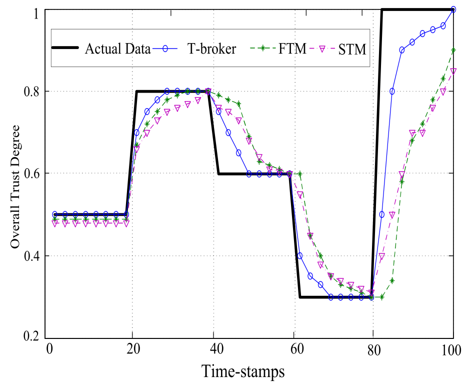

Fig. 9. Trust computing result in a random-walk profile.

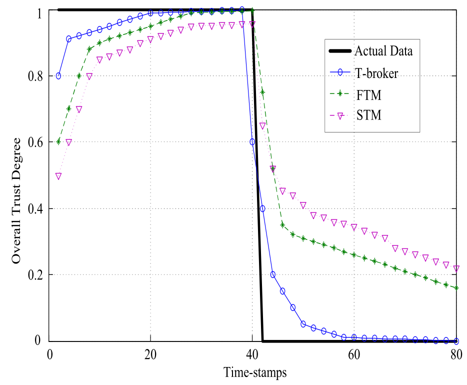

Fig. 10. Trust computing result in a cheating profile.

The random walk service periodically changes behavior. Its current normalized value of dynamic evidence e_{tk} depends on previous behavior evidence $d_{(t-1)k}$, and is defined as $e_{tk} = e_{(t-1)k} + \gamma(0,1) * U(-1,1)$, where $\gamma(0,1)$ is a real number between 0 and 1, and $U(-1,1)$ represents the random distribution from -1 to 1. In our experiment, the random walk service changes behavior every 20 time-steps, and $\gamma(0,1) = 0.8$.

2) The cheating profile models a service that turns bad once its trustworthiness (OTD) is built up. The normalized value of dynamic evidence is defined as $e_{tk} = 1$ when $t \le \Delta\lambda/2$, and $e_{tk} = 0$ otherwise, where $\Delta\lambda$ is the total number of samples.

We set the total number of observations as 100. For each dynamic value d_{tk} in d_t, we replace it with a value of random-walk or a cheating service. We can observe the OTD computing results in these two dynamic behavior profiles. Fig. 9 shows how our T-*broker* predicts the dynamic behavior of the service resource with a random walk strategy. The result shows that the three mechanisms exhibit robustness in capturing the dynamism of the service resource. However, the curve of the OTD computed by the trust mechanism in T-*broker* is highly consistent with the actual data, which indicates that the performance of the trust mechanism in T-*broker* is better than that of the FTM and STM. For a cheating service provider, its service may turn bad once its trustworthiness is built up. Fig. 10 plots the experimental results under this environment. The result shows that compared with FTM and STM, the trust mechanism in T-*broker* has better ability under such a cheating service environment. Based on a comparison of these results, the trust mechanism in this paper is more adaptable in both the random walk and cheating profiles, which indicates that the hybrid and adaptive trust calculating method represents a substantial improvement in tracking the dynamic behavior of cloud services.

VI. Conclusion and Future Work

In this paper, we present T-*broker*, a trust-aware service brokering system for efficient matching multiple cloud services to satisfy various user requests. Experimental results show that T-*broker* yields very good results in many typical cases, and the proposed mechanism is robust to deal with various number of service resources. In the future, we will continue our research from two aspects. First is how to accurately calculate the trust value of resources with only few monitored evidences reports and how to motivate more users to submit their feedback to the trust measurement engine. Implementing and evaluating the proposed mechanism in a large-scale multiple cloud system, such as distributed data sharing and remote computing, is another important direction for future research.

Appendix A
Proof of Theorem 1

Proof: To prove $s_{t_x}(r_i) < s_{t_y}(r_i)$ under condition $(t_x < t_y)$, we only need to prove that $(s_{t_x}(r_i)/s_{t_y}(r_i)) < 1$. According to Eq. (14), we can get:

$$\frac{s_{t_x}(r_i)}{s_{t_y}(r_i)} = \frac{(1-(t_x+1)^{-1}) \Big/ \sum_{t_x=1}^{\Delta\lambda}(1-(t_x+1)^{-1})}{(1-(t_y+1)^{-1}) \Big/ \sum_{t_y=1}^{\Delta\lambda}(1-(t_y+1)^{-1})}$$

$$= \frac{(1-(t_x+1)^{-1})}{\sum_{t_x=1}^{\Delta\lambda}(1-(t_x+1)^{-1})} \cdot \frac{\sum_{t_y=1}^{\Delta\lambda}(1-(t_y+1)^{-1})}{(1-(t_y+1)^{-1})}$$

because $\sum_{t_y=1}^{\Delta\lambda}(1-(t_y+1)^{-1}) = \sum_{t_x=1}^{\Delta\lambda}(1-(t_x+1)^{-1})$, thus we can get:

$$\frac{s_{t_x}(r_i)}{s_{t_y}(r_i)} = \frac{1-(t_x+1)^{-1}}{1-(t_y+1)^{-1}}$$

because $(t_x < t_y)$, we can get:

$$\begin{aligned}(t_x+1) < (t_y+1) &\Rightarrow (t_x+1)^{-1} > (t_y+1)^{-1} \\ &\Rightarrow -(t_x+1)^{-1} < -(t_y+1)^{-1} \\ &\Rightarrow 1-(t_x+1)^{-1} < 1-(t_y+1)^{-1} \\ &\Rightarrow (s_{t_x}(r_i)/s_{t_y}(r_i)) < 1\end{aligned}$$

so this condition proves Theorem 1. □

Appendix B
Proof of Theorem 2

Proof: According to Eq. (14), $\sum_{t=1}^{\Delta\lambda} s_t(r_i) = 1$. We only need prove $s_t(r_i) \in [0, 1]$. Because $t \in [1, \Delta\lambda]$, t is a positive integer. So, we can get:

$$\begin{aligned} 0 < (t+1)^{-1} < 1 &\Rightarrow 0 > (-(t+1)^{-1}) > -1 \\ &\Rightarrow 1 > (1-(t+1)^{-1}) > 0 \end{aligned}$$

that is, expression $(1-(t+1)^{-1})$ is a positive decimal. Thus, expression $\sum_{t=1}^{\Delta\lambda}(1-(t+1)^{-1})$ is also a positive decimal. Thus we can get:

$$\begin{aligned} &0 < (1-(t+1)^{-1}) < \sum_{t=1}^{\Delta\lambda}(1-(t+1)^{-1}) \\ &\Rightarrow 0 < \frac{1-(t+1)^{-1}}{\sum_{t=1}^{\Delta\lambda}(1-(t+1)^{-1})} < 1 \\ &\Rightarrow s_t(r_i) \in [0, 1] \end{aligned}$$

this condition proves Theorem 2. □

Appendix C
Proof of Theorem 3

Proof: Suppose that there are two service resources r_i and r_j. Under $\mathbb{L}(r_i) > \mathbb{L}(r_j)$, we need to prove that $\gamma_{r_i} > \gamma_{r_j}$ (or $\gamma_{r_i}/\gamma_{r_j} > 1$). For comparison under the same conditions, we suppose that the proportions of positive feedback for resources r_i and r_j are the same. That is, $q(r_i) = q(r_j)$. According to Eqs (17) and (18), we can get:

$$\frac{\gamma_{r_i}}{\gamma_{r_j}} = \frac{\rho(r_i)+q(r_i)}{\rho(r_j)+q(r_j)} = \frac{\rho(r_i)+q(r_i)}{\rho(r_j)+q(r_i)}$$

because $q(r_i) = q(r_j)$, we need to prover $\rho(r_i) > \rho(r_j)$.

$$\begin{aligned} \rho(r_i) - \rho(r_j) &= (1 - \frac{1}{\mathbb{L}(r_i)+1}) - (1 - \frac{1}{\mathbb{L}(r_j)+1}) \\ &= \frac{1}{\mathbb{L}(r_j)+1} - \frac{1}{\mathbb{L}(r_i)+1} \end{aligned}$$

because $\mathbb{L}(r_i) > \mathbb{L}(r_j)$, we can get:

$$\frac{1}{\mathbb{L}(r_j)+1} > \frac{1}{\mathbb{L}(r_i)+1}$$

this condition proves Theorem 3. □

Appendix D
Proof of Theorem 4

Proof: Suppose that there are two service resources r_i and r_j. Under $q(r_i) > q(r_j)$, we need to prove that $\gamma_{r_i} > \gamma_{r_j}$ (or $\gamma_{r_i}/\gamma_{r_j} > 1$). For comparison under the same conditions, we suppose that $\rho(r_i) = \rho(r_j)$, $\xi_{r_i} = \xi_{r_j}$. Thus, we need to prove $\gamma_{r_i} > \gamma_{r_j}$ (or $\gamma_{r_i}/\gamma_{r_j} > 1$) under condition ($\zeta_{r_i} > \zeta_{r_j}$). According to Eqs (17) and (18), we can get:

$$\frac{\gamma_{r_i}}{\gamma_{r_j}} = \frac{\rho(r_i)+q(r_i)}{\rho(r_j)+q(r_j)} = \frac{\rho(r_i)+q(r_j)}{\rho(r_i)+q(r_j)}$$

because $\rho(r_i) = \rho(r_j)$, we need to prover $q(r_i) > q(r_j)$. Because $\xi_{r_i} = \xi_{r_j}$ and $\zeta_{r_i} > \zeta_{r_j}$, we can get:

$$\begin{aligned} q(r_i)/q(r_j) &= \frac{\zeta_{r_i}}{\zeta_{r_i}+\xi_{r_i}} \Big/ \frac{\zeta_{r_j}}{\zeta_{r_j}+\xi_{r_j}} \\ &= \frac{\zeta_{r_i}}{\zeta_{r_i}+\xi_{r_i}} \Big/ \frac{\zeta_{r_j}}{\zeta_{r_j}+\xi_{r_i}} \\ &> \frac{\zeta_{r_i}}{\zeta_{r_i}+\xi_{r_i}} \Big/ \frac{\zeta_{r_j}}{\zeta_{r_i}+\xi_{r_i}} = \zeta_{r_i}/\zeta_{r_j} \\ &> 1 \end{aligned}$$

this condition proves Theorem 4. □

References

[1] M. Singhal *et al.*, "Collaboration in multicloud computing environments: Framework and security issues," *Computer*, vol. 46, no. 2, pp. 76–84, Feb. 2013.

[2] H. M. Fard, R. Prodan, and T. Fahringer, "A truthful dynamic workflow scheduling mechanism for commercial multicloud environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1203–1212, Jun. 2013.

[3] F. Paraiso, N. Haderer, P. Merle, R. Rouvoy, and L. Seinturier, "A federated multi-cloud PaaS infrastructure," in *Proc. 5th IEEE Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2012, pp. 392–399.

[4] P. Jain, D. Rane, and S. Patidar, "A novel cloud bursting brokerage and aggregation (CBBA) algorithm for multi cloud environment," in *Proc. 2nd Int. Conf. Adv. Comput. Commun. Technol. (ACCT)*, Jan. 2012, pp. 383–387.

[5] K. M. Khan and Q. Malluhi, "Establishing trust in cloud computing," *IT Prof.*, vol. 12, no. 5, pp. 20–27, Sep./Oct. 2010.

[6] K. Hwang and D. Li, "Trusted cloud computing with secure resources and data coloring," *IEEE Internet Comput.*, vol. 14, no. 5, pp. 14–22, Sep./Oct. 2010.

[7] H. Kim, H. Lee, W. Kim, and Y. Kim, "A trust evaluation model for QoS guarantee in cloud systems," *Int. J. Grid Distrib. Comput.*, vol. 3, no. 1, pp. 1–10, Mar. 2010.

[8] P. D. Manuel, S. Thamarai Selvi, and M. I. A.-E. Barr, "Trust management system for grid and cloud resources," in *Proc. 1st Int. Conf. Adv. Comput. (ICAC)*, Dec. 2009, pp. 176–181.

[9] L.-Q. Tian, C. Lin, and Y. Ni, "Evaluation of user behavior trust in cloud computing," in *Proc. Int. Conf. Comput. Appl. Syst. Modeling (ICCASM)*, Oct. 2010, pp. V7-576–V7-572.

[10] X. Li and Y. Yang, "Trusted data acquisition mechanism for cloud resource scheduling based on distributed agents," *Chin. Commun.*, vol. 8, no. 6, pp. 108–116, 2011.

[11] X. Li, H. Ma, F. Zhou, and X. Gui, "Service operator-aware trust scheme for resource matchmaking across multiple clouds," *IEEE Trans. Parallel Distrib. Syst.*, to be published, doi: 10.1109/TPDS.2014.2321750.

[12] (2014). *OPTIMIS*. [Online]. Available: http://www.optimis-project.eu/

[13] W. Fan and H. Perros, "A novel trust management framework for multi-cloud environments based on trust service providers," *Knowl.-Based Syst.*, vol. 70, pp. 392–406, Nov. 2014.

[14] N. Ghosh, S. K. Ghosh, and S. K. Das, "SelCSP: A framework to facilitate selection of cloud service providers," *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 66–79, Jan./Mar. 2015.

[15] A. Nagarajan and V. Varadharajan, "Dynamic trust enhanced security model for trusted platform based services," *Future Generat. Comput. Syst.*, vol. 27, no. 5, pp. 564–573, 2011.

[16] S. M. Habib, V. Varadharajan, and M. Muhlhauser, "A trust-aware framework for evaluating security controls of service providers in cloud marketplaces," in *Proc. 12th IEEE Int. Conf. Trust, Secur., Privacy Comput. Commun.*, Jul. 2013, pp. 459–468.

[17] T. H. Noor and Q. Z. Sheng, "Trust as a service: A framework for trust management in cloud environments," in *Web Information System Engineering* (Lecture Notes in Computer Science), vol. 6997. Berlin, Germany: Springer-Verlag, 2011, pp. 314–321.

[18] B. Rochwerger *et al.*, "The RESERVOIR model and architecture for open federated cloud computing," *IBM J. Res. Develop.*, vol. 53, no. 4, pp. 535–545, 2009.

[19] S. A. De Chaves, R. B. Uriarte, and C. B. Westphall, "Toward an architecture for monitoring private clouds," *IEEE Commun. Mag.*, vol. 49, no. 12, pp. 130–137, Dec. 2011.

[20] (2014). *RightScale*. [Online]. Available: http://www.rightscale.com/
[21] (2014). *SpotCloud*. [Online]. Available: http://www.spotcloud.com/
[22] (2014). *Aeolus*. [Online]. Available: http://www.aeolusproject.org/index.html
[23] J. Spring, "Monitoring cloud computing by layer, part 1," *IEEE Security Privacy*, vol. 9, no. 2, pp. 66–68, Mar./Apr. 2011.
[24] J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Scheduling strategies for optimal service deployment across multiple clouds," *Future Generat. Comput. Syst.*, vol. 29, no. 6, pp. 1431–1441, Aug. 2013.
[25] K. Hwang, S. Kulkarni, and Y. Hu, "Cloud security with virtualized defense and reputation-based trust management," in *Proc. 8th IEEE Int. Conf. Dependable, Autonomic, Secure Comput. (DASC)*, Dec. 2009, pp. 717–722.
[26] S. Schmidt, R. Steele, T. S. Dillon, and E. Chang, "Fuzzy trust evaluation and credibility development in multi-agent systems," *Appl. Soft Comput.*, vol. 7, no. 2, pp. 492–505, 2007.
[27] Y. L. Sun, W. Yu, Z. Han, and K. J. R. Liu, "Information theoretic framework of trust modeling and evaluation for ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 2, pp. 305–319, Feb. 2006.
[28] Z. Liang and W. Shi, "Enforcing cooperative resource sharing in untrusted P2P computing environments," *Mobile Netw. Appl.*, vol. 10, no. 6, pp. 971–983, 2005.
[29] X. Wu, "A fuzzy reputation-based trust management scheme for cloud computing," *Int. J. Digit. Content Technol. Appl.*, vol. 6, no. 17, pp. 437–445, Sep. 2012.
[30] X. Li and J. Du, "Adaptive and attribute-based trust model for service level agreement guarantee in cloud computing," *IET Inf. Secur.*, vol. 7, no. 1, pp. 39–50, Mar. 2013.
[31] X. Li, F. Zhou, and X. Yang, "Scalable feedback aggregating (SFA) overlay for large-scale P2P trust management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 10, pp. 1944–1957, Oct. 2012.
[32] W. Li and L. Ping, "Trust model to enhance security and interoperability of cloud environment," in *Proc. 1st CloudCom (Lecture Notes in Computer Science)*, vol. 5931. 2009, pp. 69–79.
[33] L. Ai, M. Tang, and C. J. Fidge, "QoS-oriented resource allocation and scheduling of multiple composite web services in a hybrid cloud using a random-key genetic algorithm," in *Proc. 18th ICONIP (Lecture Notes in Computer Science)*, vol. 7063. 2011, pp. 58–267.
[34] H. Li and M. Singhal, "Trust management in distributed systems," *Computer*, vol. 40, no. 2, pp. 45–53, Feb. 2007.
[35] J. Ma and M. A. Orgun, "Trust management and trust theory revision," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 3, pp. 451–460, May 2006.
[36] X. Li, F. Zhou, and J. Du, "LDTS: A lightweight and dependable trust system for clustered wireless sensor networks," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 6, pp. 924–935, Jun. 2013.
[37] Z. Qi, W. Zhang, and Y. Fan, "A new algorithm of weight coefficients of multiple attribute decision making," *Oper. Res. Manage. Sci.*, vol. 15, no. 3, pp. 36–39 and 65, 2006.
[38] H.-Y. Chen, "Combination determining weights method for multiple attribute decision making based on maximizing deviations," *Syst. Eng. Electron.*, vol. 26, no. 2, pp. 194–197, 2004.
[39] Y. Wang, "Using the method of maximizing deviation to make decision for multiindices," *J. Syst. Eng. Electron.*, vol. 8, no. 3, pp. 21—26, 1997.
[40] A. Whitby, A. Jøsang, and J. Indulska, "Filtering out unfair ratings in Bayesian reputation systems," in *Proc. 7th Int. Workshop Trust Agent Soc.*, New York, NY, USA, Jul. 2004, pp. 1–12.
[41] E. J. Friedman and P. Resnick, "The social cost of cheap pseudonyms," *J. Econ. Manage. Strategy*, vol. 10, no. 2, pp. 173–199, 2001.
[42] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente, "Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers," *Future Generat. Comput. Syst.*, vol. 28, no. 2, pp. 358–367, Feb. 2012.
[43] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet Comput.*, vol. 13, no. 5, pp. 14–22, Sep./Oct. 2009.
[44] F. Wuhib and R. Stadler, "Distributed monitoring and resource management for large cloud environments," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2011, pp. 970–975.
[45] P. Patel, A. H. Ranabahu, and A. P. Sheth, "Service level agreement in cloud computing," in *Proc. Cloud Workshops OOPSLA*, 2009, pp. 1–11.
[46] M. Alhamad, T. Dillon, and E. Chang, "Conceptual SLA framework for cloud computing," in *Proc. 4th IEEE Int. Conf. Digit. Ecosyst. Technol.*, Apr. 2010, pp. 606–610.
[47] D. Nurmi *et al.*, "The Eucalyptus open-source cloud-computing system," in *Proc. 9th IEEE/ACM Int. Symp. Cluster Comput. Grid*, May 2009, pp. 124–131.
[48] D. A. Swanson, J. Tayman, and T. M. Bryan, "MAPE-R: A rescaled measure of accuracy for cross-sectional subnational population forecasts," *J. Population Res.*, vol. 28, nos. 2–3, pp. 225–243, 2011.

Xiaoyong Li received the Ph.D. degree in computer science from Xi'an Jiaotong University, in 2009. He is currently an Associate Professor of Computer Science with the Beijing University of Posts and Telecommunications. He has authored over 40 papers in journals and conference proceedings. He holds five patents and three software copyrights in cloud computing, peer-to-peer computing, and other fields. His current research interests include cloud computing, network security, and trusted system. He was a recipient of the outstanding doctoral graduates in Shaanxi Province, China, in 2009, and the New Century Excellent Talents Award in University, China, in 2012.

Huadong Ma (M'99) received the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, in 1995. He is currently a Professor and the Director of the Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, and an Executive Dean of the School of Computer Science with the Beijing University of Posts and Telecommunications, China. He has authored over 100 papers and four books on these fields. His current research focuses on multimedia system and networking, Internet of Things, and sensor networks. He is a member of the Association for Computing Machinery.

Feng Zhou received the M.S. degree in computer science from the Beijing University of Posts and Telecommunications, in 1989. He is currently a Full Professor and the Director of the Center of Computer Architecture with the Beijing University of Posts and Telecommunications. He has authored or coauthored a number of papers published in journals and conference proceedings. His research interests include mobile internet, embedded computing, and communication protocols.

Wenbin Yao received the Ph.D. degree in computer Science and technology from the Harbin Institute of Technology, in 2002. He is currently a Full Professor with the School of Computer Engineering, Beijing University of Posts and Telecommunications, China. His research interests include dependable and secure computing, disaster backup and recovery, and cloud computing technologies. He is a Committee Member of Fault Tolerant Computing at the China Computer Federation and a Committee Member of Cloud Computing at China Communications Federation.