IEEE Transactions on Parallel and Distributed Systems Volume: 26 Year: 2015

Service Operator-aware Trust Scheme for Resource Matchmaking across Multiple Clouds

Xiaoyong Li, Huadong Ma, Feng Zhou and Xiaolin Gui

Abstract—This paper proposes a service operator-aware trust scheme (SOTS) for resource matchmaking across multiple clouds. Through analyzing the built-in relationship between the users, the broker, and the service resources, this paper proposes a middleware framework of trust management that can effectively reduce user burden and improve system dependability. Based on multi-dimensional resource service operators, we model the problem of trust evaluation as a process of multi-attribute decision-making, and develop an adaptive trust evaluation approach based on information entropy theory. This adaptive approach can overcome the limitations of traditional trust schemes, whereby the trusted operators are weighted manually or subjectively. As a result, using SOTS, the broker can efficiently and accurately prepare the most trusted resources in advance, and thus provide more dependable resources to users. Our experiments yield interesting and meaningful observations that can facilitate the effective utilization of SOTS in a large-scale multi-cloud environment.

Index Terms—Cloud broker, multi-cloud environment, service operator, trust scheme, resource matchmaking

1 Introduction

Regardless of the past, present or future, a crucial component of cloud computing is trust, and the problem of a trustworthy cloud service is of paramount concern for enterprises and users [1]. Users are willing to send their most sensitive data to cloud service centers, which is based on the trust relationship established between users and service providers. A lack of trust between cloud users and providers will seriously hinder the universal acceptance of clouds as outsourced computing services [2].

1.1 Motivation

Although several scholars have been attracted by the trust question of cloud service, and many studies have been carried out [2], [3], [4], [5], a universal and expanded trust scheme designed specifically for a multi-cloud computing environment is still lacking, and previous studies have some key limitations:

(1) *Few studies have focused on a trust-aware brokering framework for multi-cloud environments.* Cloud brokers can pro?vide intermediation and aggregation capabilities to enable providers to deploy their virtual infrastructures across multiple clouds [8]. The future of cloud computing will be permeated with the emergence of cloud brokers acting as intermediaries between cloud providers and users to negotiate and allocate resources among multiple data centers. Based on an integrated comparison, a number of innovative platforms have been developed for cloud brokers, such as RESERVOIR [9], PCMONS [8], RightScale [10], and SpotCloud [11] (for more information on these brokers, see **Appendix A** of the supplementary material). However, most of these platforms do not provide trust management capabilities for making trust decisions, such as selecting an optimal cloud resource to deploy a service, or optimally distributing the different components of a service among different clouds. Therefore, to increase the adoption of cloud services, a cloud broker should establish and provide trust management capacity to alleviate the worries of their users.

(2) *Few studies have focused on an expanded trust model based on dynamic service factors of a cloud resource.* From a user's perspective, trust is a comprehensive index for service guarantee and there are several trust factors in a system, i.e., security, availability, and reliability [4], [5], [12]. As observed in [12], trust is beyond security, and an expanded trust model should incorporate security, reliability, and availability factors (and other factors if possible) into a trust vector. This data can be imported from existing models to form a comprehensive trust model. This highlights the fact that the level of trust in service resources should be objectively evaluated by dynamic service operators. However, in [12], Lance et al. did not cover trust in detail, omitting numerous key issues of trust management and computing.

To the best of our knowledge, most current studies either ignore service-related operators in trust evaluation or use a unilateral context to model the trust relationship. For example, in [2], they only considered the security of services, without other trust factors. In [3], the authors only considered service operators of reliability. In [5], the authors completely ignored dynamic operators of services. A major limitation of current studies is that their schemes may lead to inaccurate trust evaluation outcomes [21].

(3) *Some schemes lack adaptability with a trust fusion calculation based on multi-dimensional service operators.* Avoiding the effect of individual favoritism on weight allocation, and confirming the weight allocation of multi-operators adaptively are very important in trust fusion calculation [7], [21]. In reality, some previous schemes are based on expert opinion to weight trust factors; however, this approach lacks adaptability and may lead to inaccurate results in trust evaluation [21]. In a recent work [3], the trustworthiness of a 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 trustworthiness of a resource i; Rc_i, Ru_i, T_i, and S_i are

- *X. Li, H. Ma and F. Zhou are with Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, P. R. China, E-mail: lxyxjtu@163.com, mhd@bupt.edu.cn, zfeng@bupt.edu.cn;*
- *X. Gui is with Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, P. R. China, E-mail: xlgui@mail.xjtu.edu.cn.*

four reliability operators of a resource; and w_{Rc}, w_{Ru}, w_T, and w_S are the weights for these operators. Although the model in [3] is a multi-operator-based evaluation scheme, it uses a manual approach to assign values to w_{Rc}, w_{Ru}, w_T, and w_S (e.g., in [3], they were all set to 0.25). Thus, this scheme lacks the adaptability to weight these reliability-related service operators.

1.2 Our Contributions

Inspired by the idea of an expanded trust evaluation approach in [12], in service operator-aware trust scheme (SOTS), we define trust as *a quantified belief by a cloud broker with respect to the security, availability, and reliability of a resource within several specified time windows.* This definition belongs to an approach based on Trusted Third Party (TTP) [6]. The broker acts as the TTP, which is composed of many registered resources. The key innovations of SOTS go beyond those of existing schemes in terms of the following aspects:

1) *A systematic trust management scheme for multi-cloud environments, based on multi-dimensional resource service operators.* SOTS evaluates the trust of a cloud resource in contrast to traditional trust schemes that always focus on unilateral trust factors of service resources. It incorporates multiple factors into a trust vector to form an expanded trust scheme to evaluate a resource. This trust scheme is more consistent with the essential attributes of a trust relationship, thus, it is more in line with the expectations of cloud users.
2) *An adaptive fused computing approach for dynamic service operators, based on information entropy theory* [23]. SOTS models the problem of trust evaluation as a process of multi-attribute decision-making, and then develops an adaptive trust evaluation approach. This adaptive fused computing approach can overcome the limitations of traditional trust schemes, in which the trusted attributes are weighted manually or subjectively.
3) *A first service, last audit (FSLA) mechanism to overcome the trust initialization problem of newly registered resources.* When a resource initially registers for business, no user has interacted with it, and consequently, information on past service operators is non-existent. In SOTS, we introduce a penalty factor-based FSLA mechanism, which can effectively remedy this problem of newly registered resources.

These designs and other specific features (e.g., a statistics-based measuring approach for multi-dimensional trust operators and a time series-based global trust predicting method) collectively make SOTS an accurate and efficient solution that can be used in multi-cloud environments.

The rest of this paper is organized as follows: Section 2 gives an overview of related work. The cloud brokering architecture is described in Section 3. Section 4 outlines the details of trust evaluation for across-cloud resources. Section 5 gives some key implementing technologies. The experimental results are presented in Section 6. Finally, Section 7 summarizes this work and suggests some future directions.

2 Related Work

Khan et al. reviewed trust in the cloud system from the user's perspective [1]. They analyzed issues of trust from a cloud users expectations, with respect to their data in terms of security and privacy. So far, many innovative trust schemes for cloud computing have been proposed by researchers, and three main classes can be identified as follows:

Reputations-based schemes. Hwang et al. suggested using a trust-overlay network over multiple data centers to implement a reputation system for establishing trust between providers and data owners [2]. Data coloring and software watermarking techniques protect shared data objects as well as massively distributed software modules. However, the authors only focused on reputation-based trust issues; they did not mention the trust problem at server level.

Self-assessment schemes. Kim et al. presented a trust evaluation model to allocate cloud resources based on providers' self-assessment [3]. Their trust model collects and analyzes reliability based on the historical server information in a cloud data center. Although the model in [3] is a multiple-attribute scheme, the authors completely ignored the real-time situation in trust relationships, which may lead to an in?complete trust decision-making outcome.

In [19], Li et al. presented a trusted data acquisition mechanism for scheduling cloud resources and satisfying various user requests. Using their trust mechanism, cloud providers can efficiently utilize their resources, as well as provide highly trustworthy resources and services to users. However, due to a lack of transparency, these self-assessment schemes [3], [19] do not completely eliminate users' trust concerns.

TTP-based schemes. Habib et al. proposed a multi-attribute trust system for a cloud marketplace [5]. This system provides means for identifying cloud providers in terms of different attributes (e.g., security, performance, compliance) that are assessed by multiple sources of trust information. However, measuring these trust attributes without giving details. Although there are some similar works available in literatures, e.g., [4], [19], which discussed the multiple-attribute issues of trust, little detail has been provided.

3 Trust-aware Brokering Architecture

3.1 Definitions, Conceptual Model and Assumptions

Referring to the description methods on "trust" in [12], [18], [22] (for related work in trust management, see **Appendix B** of the supplementary material), we first give the related definitions of "trust" that are used in SOTS.

Definition 1. **Trust of a Resource**. *Trust is a quantified belief (or a measured value) in the competence of a resource to complete a task, based on its historical service operators.*

Definition 2. **TTP-based Trust Relationship**. *A user will trust a service resource if the matchmaker (broker) states that the resource's operators will match the user's request.*

Definition 3. **Trust Evaluation Factors**. *The trustworthiness of a resource is evaluated by the broker according to multiple service operators with respect to the security, availability, and reliability of this resource within several specified time windows.*

According to Definitions 1 and 2, SOTS belongs to the TTP-based approach [6], with the broker acting as the TTP. According to Definition 3, SOTS is also an expanded trust evaluation approach [12], beyond traditional trust schemes that always focus on one-sided trust factors of service resources. The expanded trust model incorporates security, reliability, and availability factors into a trust vector. Thus, the new trust

scheme will contain data that can be imported from existing models (that is, security, reliability, availability) to form a comprehensive trust model for a multiple cloud environment.

Figure 1 shows the conceptual model of the trust-aware resource matchmaking approach. According to Definition 2, our trust scheme depends on the cloud broker, who acts as the TTP for users. The broker can evaluate each resource performance during particular time windows, thereby configuring services dynamically and distributing tasks efficiently. Whenever a new resource wants to offer its services, it must join the service network. On the client side, a user looking for a service must send a query, together with his policies, to the trusted broker. According to the trust evaluation results, the broker will select a suitable resource by applying a matching algorithm. Whenever a service resource matches, the cloud broker will distribute the user's task to the resource through its manager.

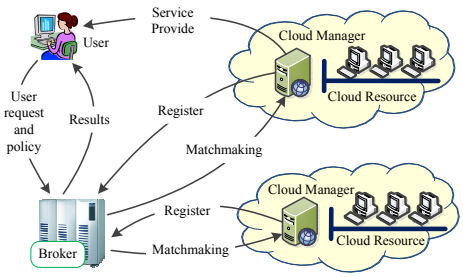

Fig. 1. Conceptual model

The underlying assumption of this TTP-based approach is that users must trust the third-party broker they decide to consult. In actual cases, these brokering systems are often managed by larger ISPs with good reputation, so the services from these ISPs should have a higher dependability. We assume that all resources have unique identities, such as the IP address, and that each cloud manager (site) can register its resources through these unique identities. This paper mainly focuses on the trust management system of server sides; thus, we also assume that each cloud site has a security mechanism to resist attacks from malicious users.

3.2 Trust-aware Brokering System Architecture

Figure 2 shows a schematic of our architecture. Conceptually, the proposed middleware architecture consists of a number of core modules, including the trusted resource matchmaking and distributing module, the adaptive trust evaluation module, the agent-based service operator acquisition module, and the resource management module, among others.

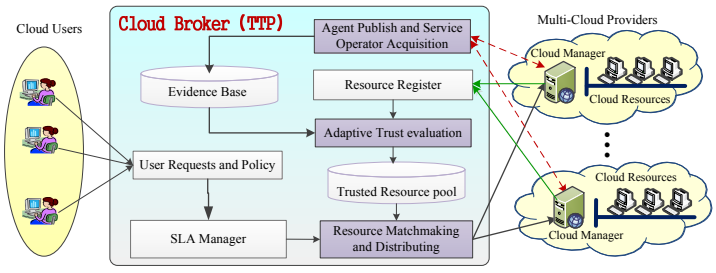

Fig. 2. A trust broker for multi-cloud environments

Adaptive trust evaluation module. This module is the core of the trust-aware cloud computing system, and is the major focus of this paper. Using this module, the broker can dynamically sort high-performance resources by analyzing the historic resource information in terms of providing highly trusted resources.

Trusted resource matchmaking and distributing module. In general, each cloud manager registers its service resources through the cloud broker. The service user negotiates with the service broker on the Service-Level Agreement (SLA) details [25]; they eventually prepare an SLA contract. According to this contract, the broker selects, and then presents highly trusted resources to users from the trusted resource pool.

Agent publish and service operator acquisition module. This module is used to monitor the usage of allocated resources in order to guarantee the SLA with the user. In interaction, the module monitors the resource operators and is responsible for getting run-time service operators. Another task of the module is to publish automatically the monitoring agents in a remote site when a computing task is assigned to the site.

Resource register module. It manages and indexes all the resources available from multiple cloud providers, and obtains information from each particular cloud resource, acting as pricing interface for users, and updating the database when new information is available.

3.3 Statistics-based Service Operator Measurement

When matchmaking a resource for users, the cloud broker must first consider whether the resource has the required capabilities (for example, CPU frequency, memory size, and hard disk capacity), and second, whether it is likely to complete the task successfully [19]. The first of these considerations can be evaluated by the resource's availability, which can determine whether a resource has the required capability or not. The second consideration mainly focuses on the reliability and security of the resource, which can be evaluated by the resource's service operators. Reliability refers to the probability of service for a given duration, and we use six operators to reflect this factor. The most basic needs of security pertain to the absence of unauthorized access to a system. We use the security levels of a service site to evaluate security [4]. The trust attributes based on dynamic service operators are summarized in Figure 3.

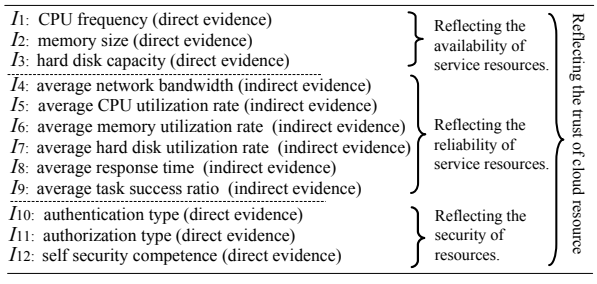

Fig. 3. Real-time and dynamic service operators

Availability measurement based on the hardware capacity of a resource. Depending on the acquisition methods, these trust factors can be divided into two types: direct operators and indirect operators. Direct operators can be obtained by some specially designed software (such as agents). An indirect operator can be obtained by simple calculation or statistics. I_1 to I_3 are direct operators; and these reflect the hardware capacity of a resource and can be obtained based

on the running profile of a machine. An example of these operators on hardware capacity is listed in Table 1.

TABLE 1
Example of the hardware capacity.

Resource	CPU frequency	Memory size	Hard disk capacity
N_1	1.90 GHz	2.0 GB	160 GB
N_2	2.40 GHz	4.0 GB	80 GB

Security measurement based on the security levels of a resource. In Fig. 3, I_{10}, I_{11} and I_{12} reflect the security capacity of a resource. The authentication type of a cloud resource is verified based on the authentication mechanism. The authorization type is verified based on the type of authorization mechanism. The self-security competence is verified by the security protection mechanism. For example, from a technical point of view, Kerberos-based authentication should have a better security level than simple password authentication, and role-based authorization should have a higher security level than simple password authorization.

To evaluate the degree of trust in a cloud resource, we should define its security levels in a quantifiable way (or by a calculable value). Referring to [4], the values for I_{10}, I_{11}, and I_{12} could be defined as positive integers 1, 2, or 3 (Table 2), reflecting elementary, intermediate, and advanced security levels, respectively. It is emphasized that instead of the above positive integers 1, 2, or 3, we can use any other number that has the property of reflecting a relative quality relationship among security levels. Our choice of the above settings is there for ease of understanding and calculation.

TABLE 2
Security level evaluation.

Security types	Security Mechanism	Value	Levels
Authentication type	Simple password	1	Elementary
	X.509	2	Intermediate
	Kerberos	3	Advanced
Authorization type	Simple password	1	Elementary
	Identity-based authorization	2	Intermediate
	Role-based authorization	3	Advanced
Self-security competence	Malware protection	1	Elementary
	Firewall protection	2	Intermediate
	Intrusion Detection System	3	Advanced

Reliability measurement based on a given time window. I_4 to I_9 are six indirect operators, which are the key indicators of reliability. The values of I_4 to I_9 are based on the statistical results within a given time window Δt. For example, for a resource performing g computing tasks within time window Δt, then

$$\begin{aligned} I_4(\Delta t) &= \sum_{i=1}^{g} B(i) \Big/ g\ , \quad I_5(\Delta t) = \sum_{i=1}^{g} C(i) \Big/ g\ , \\ I_6(\Delta t) &= \sum_{i=1}^{g} M(i) \Big/ g\ , \quad I_7(\Delta t) = \sum_{i=1}^{g} H(i) \Big/ g\ , \\ I_8(\Delta t) &= \sum_{i=1}^{g} R(i) \Big/ g\ . \end{aligned} \tag{1}$$

where $B(i)$ is the i-th measured value of the network bandwidth, $C(i)$ is the i-th measured value of the CPU utilization rate, $M(i)$ is the i-th measured value of the memory utilization rate, $H(i)$ is the i-th measured value of the hard disk utilization rate, and $R(i)$ is the i-th measured value of the response time. The operator I_9 is defined as:

$$I_9(\Delta t) = S(\Delta t)/(S(\Delta t) + U(\Delta t)) \tag{2}$$

where $S(\Delta t)$ is the number of successful interactions, and $U(\Delta t)$ is the number of unsuccessful interactions. In Eq. (2), we can adopt the traditional running result reporting mechanism (such as that used in [7]) to count $S(\Delta t)$ and $U(\Delta t)$, in which the task running result will be reported by the running site when it stops the task running (either finished or failed). However, this result reporting mechanism based on running sites will bring the problem of sites' fraud reports. To overcome this problem, in this paper, we add users' feedback information into the traditional mechanism, and four cases are taken into account:

- if the reports from both the site and the user are positive, the task is considered successful;
- if both the site and the user do not respond, the task will be considered a failure;
- if the feedback from the user is negative, the task will be processed as a failure;
- if the report from the site is positive and the user does not respond, the task is considered successful.

4 Adaptive and Efficient Trust Evaluation

4.1 Evaluation Matrix Normalization

In the process of trust evaluation, the operator set should be normalized to eliminate deviations in the results caused by each operator item's difference in the value domain.

Definition 4. *At the j-th time-stamp window Δt_j, n cloud resources are assumed to require evaluation. Thus, n refers to the total group of measurement samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_z, \cdots, \mathbf{x}_n\}$. For n resources, we can obtain a characteristic matrix:*

$$X(\Delta t_j) = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ & & \ddots & \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}. \tag{3}$$

where $1 \le z \le n$. $\mathbf{x}_z = x_{z1}, x_{z2}, \cdots, x_{zm}$.

Any row operator $x_{zk} \in I_k$ can be normalized into [0.01, 0.99] according to two cases. One case is that x_{zk} is a positive increasing value, i.e., a large value of x_{zk} is what we expect; this value covers CPU frequency, memory size, hard disk capacity, the average network bandwidth, and so on. In this case, the normalization equation is defined as follows:

$$r_{zk} = 0.01 + \frac{(x_{zk} - min(x_{zk}))(0.99 - 0.01)}{max(x_{zk}) - min(x_{zk})}. \tag{4}$$

where $max(x_{zk})$ and $min(x_{zk})$ are the maximum and minimum values of row operator x_{tk}, respectively.

The other case is that x_{zk} is a positive decreasing value, i.e., a small value of z_{tk} is what we expect; this value only covers the average response time. In this case, the normalization equation is defined as

$$r_{zk} = 0.01 + \frac{(max(x_{zk}) - x_{zk})(0.99 - 0.01)}{max(x_{zk}) - min(x_{zk})}. \tag{5}$$

where $max(x_{zk})$ and $min(x_{zk})$ are the maximum and minimum values of row operator x_{zk}, respectively.

Using Eqs. (4) and (5), each trust operator is expressed within [0.01, 0.99] and increased in a positive direction by the above-mentioned conversion. Here, the larger operator value is better. By normalizing the trust operator, we obtain a new matrix $\mathbb{R}(\Delta t_j) = (\mathbf{r}_z)_{n\times 1} = (r_{zk})_{n\times m}$, and we call $\mathbb{R}$ an evaluation matrix:

$$\mathbb{R}(\Delta t_j) = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ \mathbf{r}_n \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ & & \ddots & \\ r_{n1} & r_{n2} & \cdots & r_{nm} \end{pmatrix}. \quad (6)$$

4.2 Real-trust Trust Degree (RTD)

In the proposed scheme, RTD is used to evaluate recent cloud resource service operators, and RTD is evaluated by knowledge of a resource's quality of service. Hence, RTD is a time window-based trusted indicator for service operators, and it should be more sensitive to new operators. RTD is generated in the time window when an interaction takes place between a user and a resource.

Definition 5. *Let $\Omega = \{N_1, N_2, \cdots, N_n\}$ denote n registered resources in the broker; let $\mathcal{T}_{N_z}(\Delta t_j)$ denote the RTD of resource N_z within the j-th time window Δt_j. We define $\mathcal{T}_{N_z}(\Delta t_j)$ as follows:*

$$\mathcal{T}_{N_z}(\Delta t_j) = \mathbf{r}_z \times \{\varpi_1, \varpi_2, \cdots, \varpi_k, \cdots \varpi_m\}. \quad (7)$$

where $\mathbf{r}_z \in \mathbb{R}(z \in [1, n])$, $W = \{\varpi_1, \varpi_2, \cdots, \varpi_k, \cdots \varpi_m\}$, $\varpi_k \in [0, 1]$, and $\sum_{k=1}^{m} \varpi_k = 1$. The normalized operator vector $\mathbf{r}_z = (r_{z1}, r_{z2}, \cdots, r_{zk}, \cdots, r_{zm})$, r_{zk} is computed according to Eqs. (4) and (5). $\mathbf{r}_z$ is a sample of RTD evaluation, and it is an m-dimensional vector. ϖ_k is the weight assigned to the normalized operator r_{zk}.

Then, the key task is to compute W in Eq. (7). Obviously, the computing task for W is a problem of multi-attribute decision-making. As analyzed in Section 1, in previous studies, subjective methods were used to assign weights to these operators. This approach may lead to misinformation and could preclude an accurate evaluation of trustworthiness [21]. Thus, avoiding the effect of individual favoritism on the weight allocation of trust attributes is a key task. In this paper, we develop an adaptive trust evaluation model based on information entropy [23], which can overcome the shortage in traditional trust models, wherein the trusted attributes are weighted manually or subjectively. This innovative approach will be introduced in the following subsection.

4.3 Entropy-based and Adaptive Weight Calculation

Large-scale cloud computing environments have thousands of registered resources and tens of thousands of user requests per second, so a quick response (or effectiveness) to a user's request is a basic requirement. The information entropy approach is not only an adaptive data fusion tool, but also has a low time and space overhead in dealing with large-scale data [23], which is why we select information entropy theory as the data fusion tool for this study.

Definition 6. *The information entropy of a discrete random variable X with possible values $e_1, e_2, \cdots e_n$ is $H(X) = E(S(X))$. E is the expected value function, and $S(X)$ is the information content or self-information of X. $S(X) = \log_b(1/p(x_t))$, hence $S(X)$ is a random variable. If $p(e_t)$ denotes the probability mass function of X, then information entropy can explicitly be written as*

$$H(X) = K\sum_{z=1}^{n} p(e_z)S(e_t) = -K\sum_{z=1}^{n} p(e_z)\log_b p(e_z), \quad (8)$$

where K is a constant, and b is the base of the logarithm used. Common values of b can be 2, e, or 10. According to Eq. (8), we can give the information entropy expression of the trust decision factors, based on their self-information.

Definition 7. *Let the attribute set $I = \{I_1, I_2, \cdots, I_k, \cdots, I_m\} = \{r_{1k}, r_{2k}, \cdots, r_{zk}, \cdots, r_{nk}\}$. Each I_k denotes the row operators of attribute k, where $k = \{1, 2, \cdots, m\}$. Then, the entropy value for attribute I_k is defined as*

$$H(I_k) = -K\sum_{t=1}^{n} p(r_{zk})\ln p(r_{zk}), \quad (9)$$

where K is a constant, and $K = 1/\ln m$. The value of $p(r_{zk})$ denotes the probability mass function of each row operator r_{zk} (the computing approach of r_{zk} is shown in Eqs. (3) – (6). According to the computing approach of r_{zk} in Eqs. (4) and (5), a bigger value of r_{zk} is always better. In line with Definition 5, this larger value indicates that the resource's service operator is more trustworthy; hence, r_{zk} exhibits the trusted probability of the trust attribute I_k at a time window Δt_j. Based on this understanding of r_{zk}, and considering the entropy function of a trust attribute as a function of the probability, we define $p(r_{zk})$ as the following function:

$$p(r_{zk}) = r_{zk} \Big/ \sum_{z=1}^{n} r_{zk}\ . \quad (10)$$

After we obtain all values of $H(I_k), k = \{1, 2, \cdots, m\}$ by Eq. (9), each value in $\{\varpi_1, \varpi_2, \cdots, \varpi_k, \cdots \varpi_m\}$ (Eq. (7)) can be computed, according to the following expression:

$$\varpi_k = \gamma_k \Big/ \sum_{k=1}^{m} \gamma_k\ , \quad (11)$$

where γ_k is the entropy weight of the trust attribute I_k:

$$\gamma_k = [1 - H(I_k)]/[m - H(I_k)]. \quad (12)$$

According to Eq. (12), $\varpi_k \in [0, 1]$, and $\sum_{k=1}^{m} \varpi_k = 1$. When the entropy value $H(I_k)$ is close to 1, the tiny distinction between these entropy values will probably induce a multiplied change of entropy weights γ_k. For three entropy values, $(H(1), H(2), H(3)) = (0.999, 0.998, 0.997)$, using Eqs. (11) and (12), the entropy weight vector is (0.1667, 0.3333, 0.5000), which is clearly unreasonable. According to the entropy principle, if the entropy difference is tiny between the different attributes, then these entropy values provide the same amount of useful information. In other words, the corresponding entropy weights should show little difference. Based on this, Eq. (12) is re-defined as follows:

$$\gamma_k = [\sum_{i=1}^{m} H(I_i) + 1 - 2H(I_k)] / \sum_{j=1}^{m}(\sum_{i=1}^{m} H(I_i) + 1 - 2H(I_j)) \quad (13)$$

Theorem 1. *According to Eq.(13), if the entropy difference is tiny between the different attributes, then the corresponding entropy weights should show little difference.*

Proof: Suppose that there exists a list of $H_{I_k}, (k = 1, 2, \cdots, m)$. Due to $H_{I_k} \in [0, 1]$, we can get:

$$\sum_{j=1}^{m}(\sum_{i=1}^{m} H(I_i) + 1 - 2H(I_j)) >> \sum_{i=1}^{m} H(I_i) + 1 - 2H(I_k)$$

According to Eq.(13), a larger H_{I_k} will get a small value for γ_k, which indicates that a large entropy corresponds to a small weight. Supposing there are two entropy values H_{I_u} and H_{I_v}, we can get:

$$\frac{\gamma_u}{\gamma_v} = \frac{\sum_{i=1}^{m} H(I_i) + 1 - 2H(I_u)}{\sum_{i=1}^{m} H(I_i) + 1 - 2H(I_v)}$$

where $u, v \in (1, 2, \cdots, m)$ and $u \neq v$. Supposing that $H_{I_u} - H_{I_v} = \varepsilon(\varepsilon$ is a tiny value), then:

$$\frac{\gamma_u}{\gamma_v} = 1 - \frac{2\varepsilon}{\sum_{i=1}^{m} H(I_i) + 1 - 2H(I_v)}$$

because ε is a tiny value, and $\sum_{i=1}^{m} H(I_i) + 1 - 2H(I_v) > 1$, we can get $2\varepsilon / \sum_{i=1}^{m} H(I_i) + 1 - 2H(I_v)$ is also a tiny value. That is to say, $\frac{\gamma_u}{\gamma_v} \to 1$, which means that if the entropy difference between the different attributes is tiny, then the corresponding entropy weights should show little difference. □

As an example, for the same set of entropy values, $(H(1), H(2), H(3)) = (0.999, 0.998, 0.997)$, using improved function Eq. (13), the entropy weight vector is (0.3333,0.3333,0.3334). These results are more in line with the actual situation.

4.4 Global Trust Degree (GTD)

In previous v time windows $(\Delta t_1, \Delta t_2, \cdots, \Delta t_n)$, using Eq. (7), we can get a time series $\overline{\mathcal{D}} = \{\mathcal{T}_{N_i}(\Delta t_1), \mathcal{T}_{N_i}(\Delta t_2), \cdots, \mathcal{T}_{N_i}(\Delta t_n)\}$. GTD can be calculated by the following formula:

$$\mathcal{D}_{N_i}(\Delta t_n) = \overline{\mathcal{D}} \times \mathcal{A}^T = \sum_{j=1}^{n} (\mathcal{T}_{N_i}(\Delta t_j) \times a(\Delta t_j)), \qquad (14)$$

where $\mathcal{A} = \{a(\Delta t_1), a(\Delta t_2), \cdots, a(\Delta t_j), \cdots, a(\Delta t_n)\}$, and $\sum_{j=1}^{n} a(\Delta t_j) = 1$. $a(\Delta t_j) \in [0, 1]$ is the weights assigned to each RTD $\mathcal{T}_{N_i}(\Delta t_j)$. According to human social behavior habits, older knowledge has less impact, whereas new knowledge makes more contribution to trust decision-making [21]. Thus, we can define the $\mathcal{A} = \{a(\Delta t_1), a(\Delta t_2), \cdots, a(\Delta t_j), \cdots, a(\Delta t_n)\}$ as a time-based attenuation function:

$$a(\Delta t_j) = [1 - (1 - \lambda)^j] \Big/ \sum_{j=1}^{n} [1 - (1 - \lambda)^j], \qquad (15)$$

where $\lambda \in [0, 1]$ is an adjustable positive constant in the system, and can be tuned accordingly. According to [20], trust should meet a fundamental property: the time-based attenuation. In this study, the computing expression for GTD should reflect this fundamental property.

Theorem 2. *In* $\{a(\Delta t_1), a(\Delta t_2), \cdots, a(\Delta t_j), \cdots, a(\Delta t_n)\}$, suppose there are two time-stamps Δt_j and Δt_k. If $(j < k)$, we can get $a(\Delta t_j) < a(\Delta t_k)$, which means that Eq. (14) meets the time-based attenuation property.

Proof: We need to prove that when $(j < k)$, $(a(\Delta t_j)/a(\Delta t_k)) < 1$. According to Eq. (15), we can get:

$$\frac{a(\Delta t_j)}{a(\Delta t_k)} = \frac{[1 - (1 - \lambda)^j] \Big/ \sum_{j=1}^{n} [1 - (1 - \lambda)^j]}{[1 - (1 - \lambda)^k] \Big/ \sum_{k=1}^{n} [1 - (1 - \lambda)^k]}$$
$$= \frac{[1 - (1 - \lambda)^j]}{\sum_{j=1}^{n} [1 - (1 - \lambda)^j]} \cdot \frac{\sum_{k=1}^{n} [1 - (1 - \lambda)^k]}{[1 - (1 - \lambda)^k]}$$

because $\sum_{j=1}^{n} [1 - (1 - \lambda)^j] = \sum_{k=1}^{n} [1 - (1 - \lambda)^k]$, thus we can get:

$$\frac{a(\Delta t_j)}{a(\Delta t_k)} = \frac{1 - (1 - \lambda)^j}{1 - (1 - \lambda)^k}$$

because $\lambda \in [0, 1]$ and $j < k$, we can get:

$$(1 - \lambda)^k < (1 - \lambda)^j \Rightarrow 1 - (1 - \lambda)^j < 1 - (1 - \lambda)^k$$
$$\Rightarrow a(\Delta t_j) < a(\Delta t_k)$$

so this condition proves Theorem 2. □

For a quantitative example for values of $a(\Delta t_j)$ with different λ, see **Appendix C** of the supplementary material.

Theorem 3. *Using the proposed trust evaluation scheme, the total time complexity is not more than* $O(g) + O(mn) + O(n^2)$.

Proof: In the period of statistics-based service operator measurement (from Eq. (1) to Eq. (2)), the time complexity is g. In the period of evaluation matrix normalization (from Eq. (3) to Eq. (6)), the time complexity is $O(mn)$. In the period of RTD computing (Eq. (7)), the time complexity is $O(m)$. In the period of adaptive weight calculation for RTD (from Eq. (8) to Eq. (13)), the time complexity is $O(n^2)$. In the period of GTD computing (Eq. (14) and Eq. (15)), the time complexity is $O(n)$. Thus, the time complexity is $O(g) + O(mn) + O(m) + O(n^2) + O(n) = O(g) + O(mn) + O(n^2)$. This expression proves Theorem 3. □

Theorem 3 shows that the time complexity of the proposed trust evaluation scheme is far superior to some existing schemes, such as the fuzzy-based trust models (FTM) in [24]), whose the time complexity is $O(n^3 log_2 n)$.

5 Implementation Technologies

5.1 Overall Implementation Algorithm

We use an example to illustrate the overall algorithm for GTD calculation. Suppose that six cloud resources need to be evaluated by the broker. In a given time window (Δt_j), the broker obtains a characteristic matrix $X(\Delta t_j)$ (an example of the matrix $X(\Delta t_j)$ for the six resources can be found in **Appendix D** of the supplementary material). Then, the trust calculation steps are listed as follows:

Step 1: according to Eqs. (4) and (5), we can get an evaluation $\mathbb{R}(\Delta t_j)$ (see **Appendix D**).

Step 2: according to Eqs. (8) to (13), the weight vectors $\{\varpi_1, \varpi_2, \cdots, \varpi_{12}\}$={0.0879, 0.0403, 0.0894, 0.0742, 0.1102, 0.0586, 0.0818, 0.0748, 0.0924, 0.1092, 0.0720, 0.1092}.

Step 3: using Eq. (7), we can get the RTDs for the six resources, $\mathcal{T}_{N_1}(\Delta t_j) = 0.5332$, $\mathcal{T}_{N_2}(\Delta t_j) = 0.4194$, $\mathcal{T}_{N_3}(\Delta t_j) = 0.4177$, $\mathcal{T}_{N_4}(\Delta t_j) = 0.3653$, $\mathcal{T}_{N_5}(\Delta t_j) = 0.3575$, and $\mathcal{T}_{N_6}(\Delta t_j) = 0.7107$.

Step 4: to reflect the dynamics of RTD within multiple time windows, we need to compute GTD for the alternative resources. Suppose that there are four evaluation values for resource N_z, i.e., $\mathcal{T}_{N_z}(\Delta t_1) = 0.9, \mathcal{T}_{N_z}(\Delta t_2) = 0.85, \mathcal{T}_{N_z}(\Delta t_3) = 0.78, \mathcal{T}_{N_z}(\Delta t_4) = 0.78$. If set $\lambda = 0.9$, then according to Eq. (14), $\mathcal{D}_{N_i}(\Delta t_4) = 0.8080$.

For the GTD-based resource matchmaking algorithm, see **Appendix E** of the supplementary material.

5.2 Trust Initialization for Newly Registered Resources

When a resource initially registers for business, no user has interacted with it, and consequently, information on past service operators is non-existent. In this situation, the GTD for this new resource can not be evaluated by the proposed trust scheme.

In this work, we use the FSLA mechanism to remedy this limitation of the proposed scheme. The main idea is that the broker first evaluates the newly registered resource, based on its registered information. The FSLA mechanism will give the resource a chance to obtain users' computing tasks. In the interactions between the user and the resource, the broker will monitor its service operators, then calculate its GTD , according to the proposed trust scheme. If the commitment QoS is lower than the actual GTD that is computed by the broker, the broker will take a strict punitive step. The proposed FSLA mechanism is defined as follows:

$$\mathcal{D}_{N_z} = \rho \mathcal{D}'_{N_z}, \quad if \quad \mathcal{D}_{N_z} - \mathcal{D}'_{N_z} < \varepsilon, \tag{16}$$

where $\mathcal{D}_{N_z}$ is the actual GTD of a resource N_z, and $\mathcal{D}'_{N_z}$ is the estimated GTD, according to N_z's registered information. $\rho \in [0, 1)$ is the penalty factor, and $\varepsilon \in [0, 1)$ is the degree of tolerance set by the user. To reflect the strict punishment for deceptive service behavior, we suggest that ρ and ε should be given smaller values, such as $\rho \leq 0.5$ and $\varepsilon \leq 0.1$. This rule can ensure that the newly joined resources will provide better services, according to their commitments.

5.3 Service Operator Acquirement and Implementation

The main concern of agent-based operator acquirement involves deploying and managing the large collection of agents that are widely distributed. Moreover, another concern is in collecting and managing the monitoring data generated in a fast, dynamic, and on-demand manner. We use a collection of software agents to monitor critical resources in a multi-cloud system, and then yield time-stamped monitoring data that can be used to evaluate the RTD of a cloud resource.

Based on previous novel results [13], [14], [15], in this study, we use Java language to implement a resource monitoring system based on a Eucalyptus Platform [16]. An implementation framework of this monitoring system is shown in Figure 4. In this monitoring system, we adopt an active agent-binding mechanism [17]. When a cloud provider registers its virtual machines (VM) with the broker, each registered resource is commanded to download a monitoring agent (MA) and bind this MA with its IP. If a computing task is distributed to this resource, the MA will be activated and

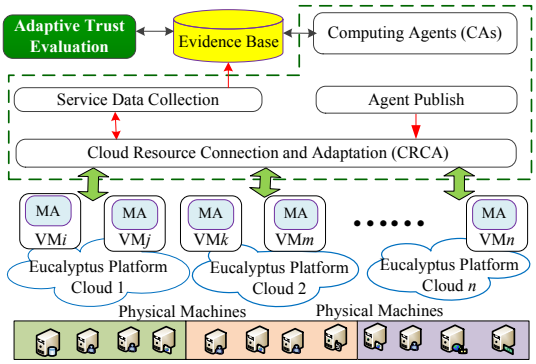

Fig. 4. An implementation framework for service operator acquisition based on Eucalyptus.

then begin its monitoring task. For acquirement of these trusted operators, two types of software agents need to be deployed: (1) MAs that run on the cloud resources, and are responsible for collecting the direct service operators of resources, such as CPU type and frequency, memory size, hard disk capacity, and others. (2) Computing agents (CAs) that run on the broker and are responsible for counting the indirect service operators, which need simple calculation and statistics, such as average response time, and average task success ratio. Using this monitoring system with the adaptive trust evaluation module, the broker can gather VMs' operators and evaluate the GTD of each VM, and then match user tasks on specific node controllers.

For performance analysis of the Eucalyptus-based platform, see **Appendix F** of the supplementary material.

6 Evaluation and Comparison

In this section, we first describe how to set up the experimental methodology in a real cloud environment, including how to deploy SOTS on the Eucalyptus-based environment and how to set the experiment configurations. Then, the experimental results are described.

6.1 Experimental Methodology

To evaluate the trust scheme based on technologies introduced in Sections 3 and 5, we set up a multiple cloud environment that is composed of three clusters (Fig. 5). Each cluster is managed by a cloud manager (3.2 GHz CPU, 4 GB memory, and 1 TB hard disk) running Ubuntu Linux 10.04 (kernel 2.6.35-24) and Eucalyptus version 1.6.1. In each cluster (cloud), the operating system running in the virtual machines is a customized Scientific Computing as a Service (SCaaS) [26]. Each cloud under test is fully based on the Eucalyptus framework and the KVM hypervisor [16].

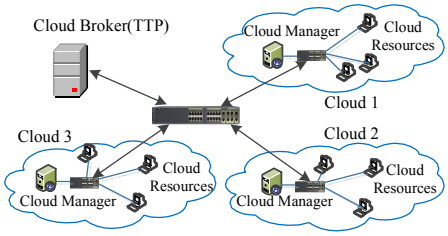

Fig. 5. Multiple cloud experimental environment.

In Fig. 5, machines in each cluster act as VM providers, in which an agent-based service operator acquisition module is deployed. A separate machine acts as the trust manage?ment server (cloud broker) where the core functional modules of the broker are deployed, including a trusted resource matchmaking and distributing module, an adaptive trust evaluation module, and a resource management module. We have designed several performance mechanisms for a comprehensive trust evaluation scheme. Due to the restrictions of paper length, we mainly evaluate the performance of SOTS based on the following two aspects:

- Accuracy is used to check whether the proposed scheme and its related algorithms can accurately and consistently provide trust calculation;
- Efficiency is used to evaluate the overhead and the average job failure rate (AJFR) of the proposed scheme;

User request. In the experimental environment, there are nearly 100 VMs in the resource pool of the cloud broker system (we deployed more than 30 VMs in each cluster). According to Algorithm 1 in **Appendix E**, our resource matchmaking approach should be "trust with cost." The user's request contains the job descriptions; namely, Job ID, minimum GTD required, and cost limits. Considering the job requirements, a resource is selected from a resource pool that has more than the minimum GTD given by the user.

TABLE 3
Classification threshold for VMs in the experiments

Types of VMs	CPU(G)	ART(sec)	ATSR(%)	AET	AOT	SEC
H	3.0	2	90-100	3	3	3
N	2.4	5	80-89	2	2	2
L	1.7	10	60-79	1	1	1
M	1.0	15	0-59	0	0	0

Types of VMs. To reduce complexity, in the initial stage of the experimental environment, we mainly observe the results according to the following 6 key operators: CPU frequency (CPU), average response time (ART), average task success rate (ATSR), authentication type (AET), authorization type (AOT), and self-security competence (SEC). Types of VMs in the resource pool and the classification threshold are listed in Table 3, including high trusted node (H), normal trusted node (N), low trusted node (L) and malicious node (M).

6.2 Accuracy Evaluation

Some service operators are more sensitive to users requirements, including average response time, average task success rate and resource security levels. Relative to a given user-sensitive trust operator, an accuracy trust scheme should be robust enough to detect operators with a smaller value. Hence, we observe the detecting capacity of the proposed scheme for low-value operators.

Using 10 time windows (each Δt=600 sec), we gather 1500 training samples from three clusters. For purposes of comparison, we also implement two other typical trust evaluation schemes: the weighted average trust model (WATM) [3] and the multi-dimensional trust model (MDTM) [21]. WATM is similar to Kim's model [3], and uses an average approach to assign weights to trust attributes: $\mathcal{G}(N_z) = (\sum_{k=1}^{m} r_{zk})/m$, where $\mathcal{G}(N_z)$ is the RTD of resource N_z, r_{zk} is the normalized value of the k-th operator at time window Δt, and m is the total number of trust operators. Obviously, the WATM is a subjective measurement approach; it assigns equal weights to each trust operator.

In MDTM, the weights for these trust operators are assigned by a weighted moving average and ordered weighted averaging (WMA-OWA) combination algorithm [21]: $\mathcal{F}(N_z) = \{w_1, w_2, \cdots, w_m\} \circ \{x_{z1}, x_{z2}, \cdots, x_{zm}\}$, where $\mathcal{F}(N_z)$ is the RTD of resource N_z, operator $\circ$ is the WMA-OWA compose operator, $x_{zk}(k \in [1, m])$ is the normalized value of the k^{th} operator, and $\{w_k, k \in [1, m]\}$ is the weight vector for these trust operators. In MDTM, the vector $\{w_k\}$ is computed by the WMA-OWA combination operator.

If a resource has a low-value user-sensitive trust operator (such as less than 0.5 for average task success rate), this resource should not be selected as a service provider. Within n evaluated samples $\{r_{zk}\}, z \in [1, n], k \in [1, m]$, let $\overline{r}_{zk} = \sum_1^n r_{zk}/n$. If $\mathcal{D}_{N_z} > 0.5$ and $r_{zk} > \overline{r}_{zk}$, this resource is a hit resource. The number of hit resources can be used to reflect the accuracy of a trust scheme.

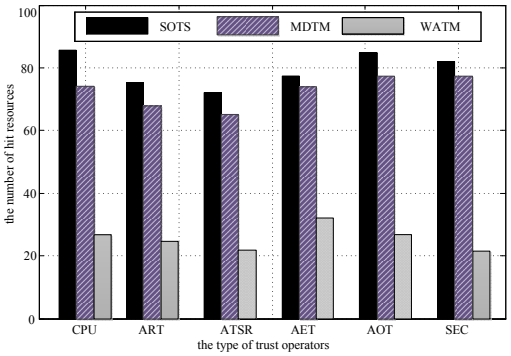

Fig. 6. Number of hit VMs falling into the top 25% values of the six trust operators.

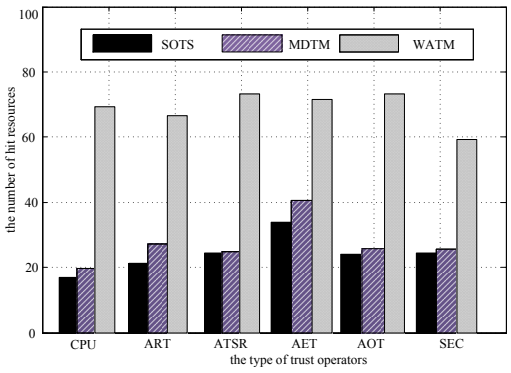

Fig. 7. Number of hit VMs fallen into the bottom 25% values of the six trust operators.

In Figure 6, the SOTS is shown to have the most number of hit VMs in six cases, and WATM is shown to have the least. MDTM has a similar number of hit resources with our scheme. Moreover, the number of hit VMs of WATM is only one third of that of our scheme and of MDTM, which indicates that adaptive weighting approaches (SOTS and MDTM) are far superior to the subjective weighting approach (WATM). Further observing the results under an opposite situation, the number of hit VMs fallen into the bottom 25% values of the six trust operators and the comparison results

are shown in Figure 7. The number of hit VMs of WATM is shown to be much higher than that of our scheme and of MDTM, which implies that the subjective approach makes more misjudgments in trusted resource selection. This finding further verifies that the proposed trust scheme is dependable in resource matchmaking.

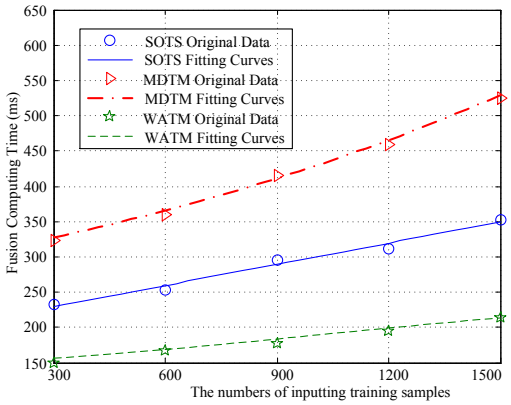

Fig. 8. FCT in cases of different number of training samples

However, from Figures 6 and 7, MDTM is shown to have similar results to our scheme. Hence, we use Figure 8 to explain the reason that we adopt the information entropy theory as the chosen fusion tool for trust operators instead of oth?er tools, such as the WMA-OWA combination algorithm. Large-scale cloud computing environments have thousands of registered resources, thereby requiring a quick response to a user's requests. We use fusion computing time (FCT) to evaluate this indicator. The computing process is carried out by a computer with 3.2 GHz CPU, 4 GB of memory. In Figure 8, the FCT of WATM is between 150 to 210 *ms* with different numbers of training samples, which demonstrates that WATM has the fastest computing speed among the three models. However, the changing trend of MDTM is the largest among the three models. In comparing these results in Figure 8, our scheme is shown to need less FCT than that of MDTM. Hence, compared with MDTM, SOTS has a better quick-response capability. By comprehensive analysis of the results in Figures 6, 7, and 8, SOTS has both accuracy and rapidity, compared with the other two schemes.

6.3 Efficiency Evaluation

According to the VM types listed in Table 3, we set up two typical resource scenarios: S1 and S2, which are described in Table 4. S1 considers the community to be a trusted resource scenario with 90% trusted VMs. S2 considers the community to be a malicious resource scenario with 20% malicious nodes and 20% low trusted VMs.

TABLE 4
Two types of network environment

Scenarios	H	N	L	M
S1	80%	10%	3%	2%
S2	50%	10%	20%	20%

Based on AJFR, we evaluate the efficiency of SOTS with respect to the resource matchmaking problem. In these experiments, once the job is recorded as a failure, it is resubmitted until all the jobs are successfully executed. The AJFR is defined as follows [19]:

$$AJFR = [(\delta_d + \beta_r) \times 100\%]/(\delta_{total} + \beta_r), \quad (17)$$

where δ_d is the number of delayed jobs, and δ_{total} is the total number of submitted jobs. β_r is the number of resubmitted jobs. A good resource matchmaking mechanism should have a smaller value of AJFR.

We adopt three resource matchmaking mechanisms in this group of experiments: (1) strict cost-based matchmaking mechanism (SCSM), (2) random matchmaking mechanism (RSM) and (3) trust-based matchmaking mechanism (includ?ing our scheme, WATM, and MDTM). RSM is the simplest, and is used to choose a resource randomly from the idle resource pool when a user's request is detected. SCSM is a greedy method; it only considers user cost (UC), with?out considering resource trust. The trust-based matchmaking mechanism sorts high-performance resources by analyzing the VM history for providing highly trusted resources dynamically when user requests arise.

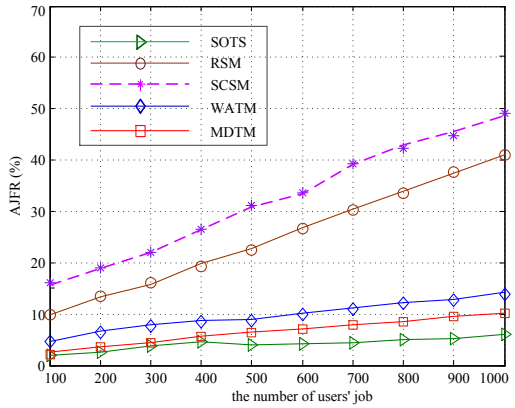

Fig. 9. Experimental results under condition S1.

Figure 9 shows the AJFR as computed by the five schemes under a high percentage of trusted VMs. The total percentage of high trusted VMs is 80%; the total percentage of normal trusted VMs is 10%; the total percentage of malicious VMs and low trusted VMs is 5%. This resource scenario considers the cloud computing environment to be a relatively good community with few malicious resources. Figure 9 further shows the growth in the number of users' jobs for SCSM and RSM, with average values of AJFRs of 23.6% and 34.2%, respectively. However, the three trust-based matchmaking schemes, namely, SOTS, FTM and WATM, have relatively stable performance, with average AJFRs from 4.2% to 9.7%, which reflects that the three schemes are robust when facing few malicious resources. Thus, from a practical point of view, these three schemes can all meet the demand.

Figure 10 shows the experimental results as computed by the five mechanisms under a high percentage of malicious VMs and low trusted VMs. The percentage of high trusted VMs is only 50%; the percentage of normal trusted VMs is 10%; the total percentage of malicious VMs and low trusted VMs is 40%. This resource scenario considers the multi-cloud computing environment to be a bad community, in which the percentage of malicious resources and low trusted VMs reaches nearly half of the total resource. When the percentage of low trusted resource increases from 5% to 20%, the

AJFR obtained by SCSM shows notable dynamic fluctuation around 60%. For RSM, its AJFR quickly increases from 20.9% to 70.5%. For WATM, its average value of AJFR is 32.9%, which reflects a significant decline in the performance of this resource matchmaking scheme. However, the other two trust-based matchmaking schemes, SOTS and MDTM, have relatively stable performance with average AJFRs from 9.8% to 18.9%, which demonstrates that these two schemes can also be robust when facing a large number of malicious resources. However, compared with MDTM, the trust scheme proposed here shows better choices under experimental condition S2.

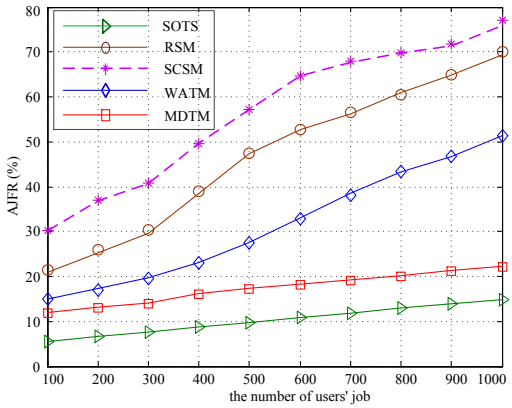

Fig. 10. Experimental results under the condition S2.

7 Conclusion and Future Work

In this work, we propose SOTS for trustworthy resource matchmaking across multiple clouds. We have shown that SOTS yields very good results in many typical cases. How?ever, there are still some open issues we can apply to the current scheme. First, we are interested in combining our trust scheme with reputation management to address concerns in users' feedback. A universal measurement and quantitative method to assess the security levels of a resource is another interesting direction. Evaluation of the proposed scheme in a larger-scale multiple cloud environment is also an important task to be addressed in future research.

Acknowledgments

The authors would like to appreciate the associate editors and the anonymous reviewers for their insightful suggestions to improve the quality of this paper. This work is supported by the National Nature Science Foundation of China (No. 61370069, No. 61320106006), Beijing Natural Science Foundation (No. 4111002), Fok Ying Tung Education Foundation (No. 132032) and Program for New Century Excellent Talents in University (No. NCET-12-0794).

References

[1] K. M. Khan, Q. Malluhi, "Establishing Trust in Cloud Computing", *IEEE IT Professional*, vol. 12, no. 5, 2010, pp. 20-27.
[2] K. Hwang, D. Li, "Trusted Cloud Computing with Secure Resources and Data Coloring", *IEEE Internet Computing*, vol. 14, no. 5, 2010, pp. 14-22.
[3] H. Kim, H. Lee, W. Kim, Y. Kim, "A Trust Evaluation Model for QoS Guarantee in Cloud Systems", *International Journal of Grid and Distributed Computing*, vol.3, no.1, pp. 1-10, 2010.
[4] P. D. Manuel, S. Thamarai Selvi, M. I. A. E. Barr, "Trust management system for grid and cloud resources", *Proc. of the First International Conference on Advanced Computing* (ICAC 2009), 2009, 13-15 Dec, pp. 176-181.
[5] S. M. Habib, S. Ries, and M. Muhlhauser, "Towards a Trust Management System for Cloud Computing", *Proc. of IEEE TrustCom-11/IEEE ICESS-11/FCST-11*, pp. 933-939, 2011.
[6] N. Dragoni, "A Survey on Trust-Based Web Service Provision Approaches", Proc. of the 2010 Third International Conference on Dependability, pp. 83-99, 2010.
[7] Z. Liang and W. Shi, "A reputation-driven scheduler for autonomic and sustainable resource sharing in Grid computing" *Journal of Parallel and Distributed Computing*, vol. 70, no. 2, pp.111-125, 2010.
[8] S. A. de Chaves, R. B. Uriarte, C. B. Westphall, "Toward an Architecture for Monitoring Private Clouds", *IEEE Communications Magazine*, vol. 49, no. 2, pp. 130-137, 2011.
[9] S. Clayman, A. Galis, C. Chapman, et al., "Monitoring Service Clouds in the Future Internet". *Future Internet Assembly*, pp. 115-126, 2010.
[10] RightScale, http://www.rightscale.com/.
[11] SpotCloud, http://www.spotcloud.com/.
[12] L.J. Hoffman, K. Lawson-Jenkins, and J. Blum, "Trust beyond Security: An Expanded Trust Model", *Communications of the ACM*, vol. 49, no. 7, pp. 95-101, 2006.
[13] K. R. Jackson, K. Muriki, L. Ramakrishnan, et al., "Performance and cost analysis of the Supernova factory on the Amazon AWS cloud". *Scientific Programming*, vol. 19, no. 2-3, pp. 107-119, 2011.
[14] D. Ghoshal, R. S. Canon, and L. Ramakrishnan. "I/O performance of virtualized cloud environments". Proc. of the second international workshop on Data intensive computing in the clouds. pp.71-80, 2011.
[15] J. Shafer. "I/O virtualization bottlenecks in cloud computing today". Proc. of the 2nd conference on I/O virtualization, pp. 1-7, 2010.
[16] R. Wolski, C. Grzegorczyk, G. Obertelli, et al., "The Eucalyptus Open-Source Cloud-Computing System ", Proc. of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 124-131, 2009.
[17] I. Legrand, R. Voicu, C. Cirstoiu, C. Grigoras, L. Betev, A. Costan, "Monitoring and control of large systems with MonALISA". *Commun. ACM* vol. 52, no. 9, pp. 49-55, 2009.
[18] F. Azzedin and A. Ridha. "Feedback Behavior and Its Role in Trust Assessment for Peer-to-Peer Systems". *Telecommunication Systems*, vol. 44, no. 3-4, pp. 253-266, 2010.
[19] X. Li, Y. Yang. "Trusted Data Acquisition Mechanism for Cloud Resource Scheduling Based on Distributed Agents". *China Communications*, vol.8, no. 6, pp,108-116, 2011.
[20] X. Li, F. Zhou, X. Yang, "Scalable Feedback Aggregating (SFA) Overlay for Large-Scale P2P Trust Management". *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 10, pp. 1944-1957, 2012.
[21] X. Li, F. Zhou, X. Yang, "A Multi-Dimensional Trust Evaluating Model for Large-scale P2P Computing", *Journal of Parallel and Distributed Computing*, vol.71, no.6, pp.837-847, 2011.
[22] X. Li, F. Zhou, J. Du, "LDTS: A Lightweight and Dependable Trust System for Clustered Wireless Sensor Networks", *IEEE Transactions on Information Forensics and Security*, vol.8, no.6 pp. 924-935, 2013.
[23] H. Zhou, G. Zhang, G. Wang, "Multi-objective decision making approach based on entropy weights for reservoir flood control operation", *Journal of Hydraulic Engineering*, Vol. 38, no. 1, pp. 100-106, 2007.
[24] S. Stefan, S. Robert, "Fuzzy trust evaluation and credibility development in multi-agent systems", *Applied Soft Computing*, vol. 41, no. 7, pp. 492-505, 2007.
[25] M. Alhamad, T. Dillon, E. Chang. "Conceptual SLA Framework for Cloud Computing", Proc. 4th IEEE International Conference on Digital Ecosystems and Technologies, pp.606-610, 2010.
[26] P. Saripalli, C. Oldenburg, B. Walters, et al., "Implementation and Usability Evaluation of a Cloud Platform for Scientific Computing as a Service (SCaaS)", Proc. of the 4th IEEE Int. Conf. on Utility and Cloud Computing, pp. 345-354, 2011.