International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Optimization of Trust Management Scheme for Multi Cloud Environment

Ruchira Tare[1], Rupali Pandharpatte[2]

[1]M.E. Student Department of Computer Engineering, KJ College of Engineering Management & Research, Savitribai Phule Pune University, India

[2]Assistant Professor, KJ College of Engineering Management & Research, Savitribai Phule Pune University, India

Abstract: ***Cloud computing is the use of computing resources such as hardware and software that are delivered as a subscription-based service and on demand services over a network. In cloud computing environments, there are two players: cloud providers and cloud users. Users always want to send their most sensitive data to cloud service centers, which is based on the trust relationship established between users and service providers. So we require such a middleware framework of trust management that can effectively reduce user burden and improve system dependability. To increase the adoption of cloud services, a cloud broker should establish and provide trust management capacity to alleviate the worries of their users. Using SOTS [Service Operator-aware Trust Scheme], the broker can efficiently and accurately prepare the most trusted resources and thus provide more dependable resources to users. Traditionally cloud providers provide assurances by specifying technical and functional descriptions in Service level agreements (SLAs) for the services they offer. But the customers are not sure whether they can identify trustworthy cloud providers only based on its SLA. We address SOTS for trustworthy resource matchmaking across multiple clouds. In this work we can facilitate the effective utilization of SOTS in a large scale multi-cloud environment by using GTD based resource matchmaking algorithm and FSLA mechanism.***

Keywords: Cloud broker, Multi-cloud environment, Trust Scheme, Resource Matchmaking.

1. Introduction

Oriented by requirement of trust management in multiple cloud environments, 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 environments, in which the Broker acts as a middleware for cloud trust management and service matching. Then, 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, 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, Broker uses a lightweight feedback mechanism, which can effectively reduce networking risk and improve system efficiency.

The 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.Amiddleware 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.

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 cloudusers 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:

2.1 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.

2.2 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 incomplete 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.

2.3 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. Proposed Work

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.

3.1 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.

3.2 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; 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.

3.3 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.

Advantages

- The Broker is aware of the resources seeking and providing with the matchmaking framework.
- It makes the resource availability with using security key for sharing the content with highest security.

4. Simulation Results

4.1 CloudSim Extensions

CloudSim is a scalable, open-source simulation tool offering features like support for modeling and simulation of large-scale Cloud computing infrastructures, including datacenters, brokers, hosts, and virtual machines (VMs) on a single host. In addition, the support for custom developed scheduling and allocation policies in the simulation made CloudSim an attractive tool for Cloud researchers. In our simulation environment, CloudSim is used to model large-scale and heterogeneous Cloud providers. This allows us, for the purpose of evaluation, to easily configure the amount of Cloud provider resources accessible by the broker. Nevertheless, some CloudSim extensions were needed to allow the dynamic creation, destroying and monitoring of the VMs.

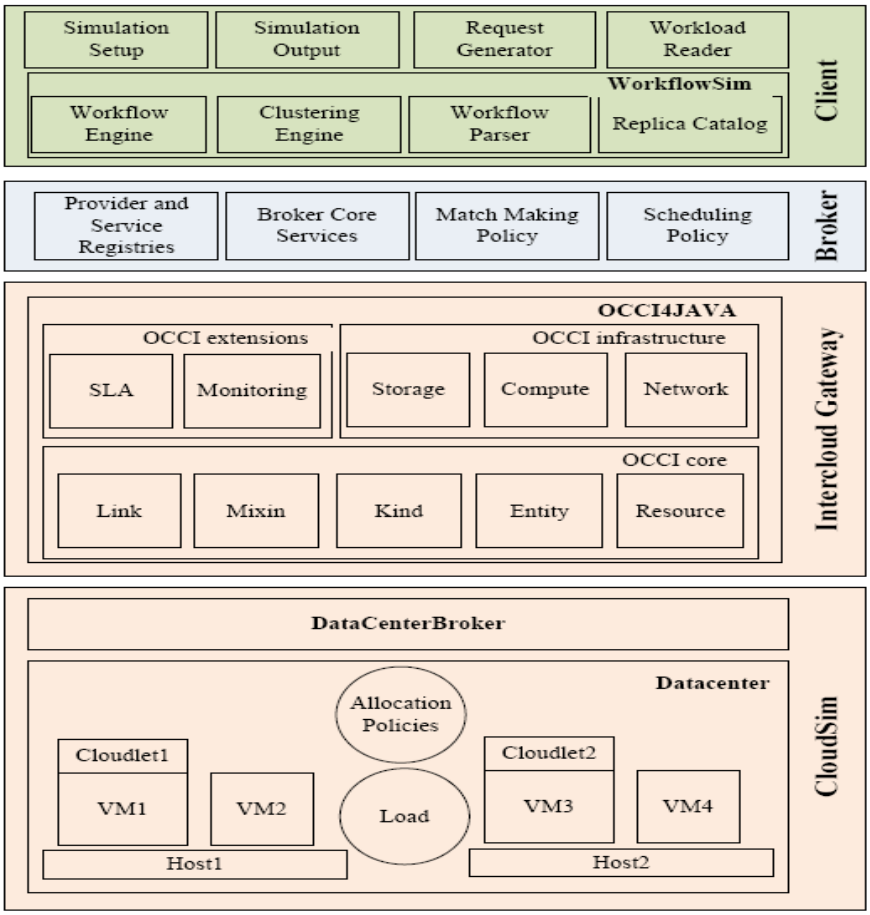

Figure 1: Simulation Environment

4.2 Cloud Service Broker Implementation

We implemented the core broker services including the SLA manager, deployment manager, the match maker, and the monitoring manager as Java classes included in the Cloud service broker package. The implemented match maker functionality of the broker is extensible enough to permit the easy integration and evaluation of different resource matching policies. Furthermore, two persistence classes, named ServiceRegistry and ProviderRegistry, are used to store and query all the service and provider data stored using the previously presented ontologies during the simulation. The ontologies are implemented in the classes ServiceRequest and Provider, which are the abstractions of a composite service request and a Cloud provider respectively.

4.3 Intercloud Gateway Implementation

In order to simulate the Intercloud gateway component serving as standard service frontend for Cloud providers, we implemented, based on the open source Java implementation for OCCI called OCCI4JAVA [120], an OCCI frontend for CloudSim. In this way, the entire communication between broker and providers is forwarded to the native CloudSimDatacenterBroker class through standard OCCI-interfaces.The use of an OCCI-based Intercloud gateway allows us to model a multi-Cloud infrastructure consisting of interoperable Clouds mediated by a Cloud service broker.

4.4 Request Generator

The simulation-based evaluation of the broker requires the modeling of realistic service requests to achieve valuable evaluation results. Thus, we implemented a service RequestGenerator helper class that continuously generates synthetic computing service requests with different VM types at a configurable rate. The configuration of the VMs is similar to the configuration of the compute instances provided by current commercial Clouds.

4.5 Workload Reader

In order to have more realistic simulation results, we included a WorkloadReader class to import the service requests and resource workloads from real workload traces like the Grid workload archive or the PlanetLab trace data . The imported trace data is used then to dynamically generate the CloudSim Cloudlets, which model the workload on the requested VMs. The use of Grid traces is justified by the lack of public accessible real Cloud traces.

The client provides Cloud users with an interactive user interface to submit their service requests to the broker by describing the functional and non-functional service requirements. Moreover, the user is able to manage and monitor the service after its deployment through a single management console. The client includes support the deployment of workflow applications on multi-Cloud. It delivers the workflow tasks to the underlying Cloud service

broker and takes care of their dependencies. Additionally, a replica catalog is used to manage data replicas.

5. Equations

Let resource has the required capabilities
I1: CPU frequency (direct evidence)
I2: memory size (direct evidence)
I3: hard disk capacity (direct evidence)

5.1 Reliability measurement based on a given time window

For example, for a resource performing g computing tasks within time window
$I1(\Delta t) = g \Sigma i=1 CPU(i)/g$,
$I2(\Delta t) = g \Sigma i=1 MEM(i)/g$,
$I3(\Delta t) = g \Sigma i=1 HDD(i)/g$,

the i-th measured value of the response time. Measure defined as:
$M(\Delta t) = S(\Delta t)/(S(\Delta t) + U(\Delta t))$

5.2 Entropy-based and Adaptive Weight Calculation

The information entropy of a discrete random variable X with possible values e1, e2, · · · en is H(X) = E(S(X)). E is the expected value function, and S(X) is the information content or self-information of X.
$H(X) = Kn\Sigma z=1\ p(ez)S(et) = -K\ n\ \Sigma z=1\ p(ez)\ logb\ p(ez)$,

Global Trust Degree (GTD)
$DNi(\Delta tn) = D \times AT = n\ \Sigma j=1\ (H(X) \times M(\Delta t))$,
where A = {a(Δt1), a(Δt2), · · · , is the weights assigned to each real trust of Reliability measurement

DSA is a Signature Scheme with Appendix. This means the that the message must be presented to the verifier function. This is in contrast to a Signature Scheme with Recovery. In a recovery system, the message is folded into the signature, so the message does not have to be sent with the signature. The verification routine will extract the message from the signature in a recovery system.

Key Generation
A DSA key is generated as follows [12]. Below, the size of q is fixed by FIPS 186 at 160 bits. Though the original FIPS 186 specification [7] specifies p between 512 to 1024 bits inclusive, FIPS 186-2 [11] fixes p at 1024. This means that some libraries enforce a bit size of 1024 at step three.

1. Select a prime number q such that $2^{159}<q< 2^{160}$
2. Choose t so that $0 \leq t \leq 8$
3. Select a prime number p such that $2^{511+64t}<p< 2^{512+64t}$ with the additional property that q divides $(p-1)$
4. Select a generator α of the unique cyclic group of order q in Z^*_p
5. To compute α, select an element g in Z^*_p and compute $g^{(p-1)/q} \bmod p$
6. If $\alpha = 1$, perform step five again with a different g
7. Select a random a such that $1 \leq a \leq q-1$
8. Compute $y = \alpha^a \bmod p$

The public key is (p, q, α, y). The private key is a. We usually encounter the private key specified as x.

Message Signing
To sign a document of arbitrary size using an appendix scheme, two steps occur:
- hash the document
- decrypt the hash of the document as if it were an instance of ciphertext using the private key

In DSA, the details of signing the binary message m (document) of arbitrary length are as follows [12]. Notice that we are signing a binary message (there is no notion of a string at this level), and the message can be any length. Because the message can be any length, the message is digested with a hash function — $h(m)$.

1. Generate a random per-message value k such that $0 <k<q$
2. Compute $r = (\alpha^k \bmod p) \bmod q$
3. If r = 0, perform step one again with a different k
4. Compute $k^{-1} \bmod q$
5. Calculate $s = k^{-1}\{h(m) + ar\} \bmod q$
6. If s = 0, perform step one again with a different k

The signature on m is (r, s). Message m and (r, s) should be sent to the verifier. We need to observe that both r and s are 20 bytes, since a modular reduction is being performed (steps 2 and 5) using q, a 160 bit value. (which use the IEEE P1363 signature format) and Java (which uses a DER encoding of a signature).

Message Verification

To verify a document of arbitrary size using an appendix scheme, three steps occur:
- hash the document
- encrypt the previously generated document hash (from step 2 of Message Signing process) using the signer's public key
- verify the recovered hash from step one of the Message Verification process matches the calculated hash from step two of the Message Verification process

The short story of the above is we are comparing our calculated hash of the document with the signer's calculated hash of the document after we remove the signer's encryption operation. The DSA details are. Below, recall that (r, s) is the signature on binary message m, with $h(m)$ digesting the arbitrary length message.

1. Obtain the public key (p, q, α, y)
2. Verify $0 <r<q$ and $0 <s<q$ (reject the signature otherwise)
3. Compute $w = s^{-1} \bmod q$
4. Compute $u_1 = w \bullet h(m) \bmod q$
5. Compute $u_2 = rw \bmod q$
6. Compute $v = (\alpha^{u_1} y^{u_2} \bmod p) \bmod q$

The signature is valid if and only if $v = r$.

References

[1] 2 MASS. 2MASS at IPAC.[Online], 2014.http://www.ipac.caltech.edu/ 2mass/(accessed:2014-03-20).

[2] Lifeng Ai, Maolin Tang, and Colin J. Fidge. Partitioning composite web services for decentralized execution using a genetic algorithm. Future Generation Comp. Syst., 27(2):157–172, 2011.
[3] G. A. Akerlof. The market for lemons: Quality uncertainty and the market mechanism. The Quarterly Journal of Economics, 84(3):488–500, 1970.
[4] MohammadAlrifai, Thomas Risse, Peter Dolog, and Wolfgang Nejdl. ―A scalable approach for qos-based web service selection". In George Feuerlicht and Winfried Lamersdorf, editors, Service-Oriented Computing ICSOC 2008 Workshops, volume 5472 of Lecture Notes in Computer Science, pages 190–199. Springer Berlin Heidelberg, 2009.
[5] Amazon Elastic Compute Cloud. [Online], 2014.http://aws.amazon.com/ ec2/ (accessed: 2014-03-20).
[6] Amazon Simple Storage Service. [Online], 2014. http://aws.amazon.com/ (accessed: 2014-03-20).
[7] D. Ardagna, E. Di Nitto, P. Mohagheghi, S. Mosser, C. Ballagny, F. D'Andria, G. Casale, P. Matthews, C.-S.Nechifor, D. Petcu, A. Gericke, and C. Sheridan.―Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds". In Modeling in Software Engineering (MISE), 2012 ICSE Workshop on, pages 50–56, June 2012.
[8] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. ―Above the Clouds: A Berkeley View of Cloud Computing". Technical report, University of California at Berkeley, February 2009.
[9] John Asker and Estelle Cantillon.―Properties of scoring auctions". The RAND Journal of Economics, 39(1):69–85, 2008.
[10] Microsoft Azure Platform. [Online], 2014.http://www.microsoft.com/windowsazure/ (accessed: 2014-03-20).

Author Profile

Ruchira Tare is pursuing her Masters of Engineering in the Computer Networks, Computer Department, KJ Collegeof Engineering Management & Research, and Savitribai Phule Pune University. She received Bachelor of Engineering degree in Computer Science & Engineering from Dr.Babasaheb Ambedkar Marathwada University, Aurangabad, India.