Journal of ICT, 13, 2014, pp: 21–36

STATISTICALLY CONTROLLED ROBUST TRUST COMPUTING MECHANISM FOR CLOUD COMPUTING

[1]Mohamed Firdhous, [2]Osman Ghazali & [3]Suhaidi Hassan
[1] Faculty of Information Technology, University of Moratuwa, Sri Lanka
[2,3]School of Computing, Universiti Utara Malaysia, Malaysia

firdhous@uom.ik; osman@uum.edu.my; suhaidi@uum.edu.my

ABSTRACT

Quality of service plays an important role in making distributed systems. Users prefer service providers who meet the commitments specified in the Service Level Agreements to these who violate them. Cloud computing has been the recent entrant to the distributed system market and has revolutionized it by transforming the way the resources are accessed and paid for. Users can access cloud services including hardware, development platform and applications and pay only for the usage similar to the other utilities. Trust computing mechanisms can play an important role in identifying the right service providers who would meet the commitments specified in the Service Level Agreements. Literature has reported several trust computing mechanisms for different distributed systems based on various algorithms and functions. Almost all of them modify the trust scores monotonously even for momentary performance deviations that are reported. This paper proposes a trust computing mechanism that statistically validates the attribute monitored before modifying the trust scores using a hysteresis-based algorithm. Hence the proposed mechanism can protect the trust scores from changes due to momentary fluctuations in system performances. The experiments conducted show that the trust scores computed using the proposed mechanism are more representative of the long-term system performance than the ones that were computed without the validation of the inputs.

Keywords: Trust management, system performance, system fluctuations.

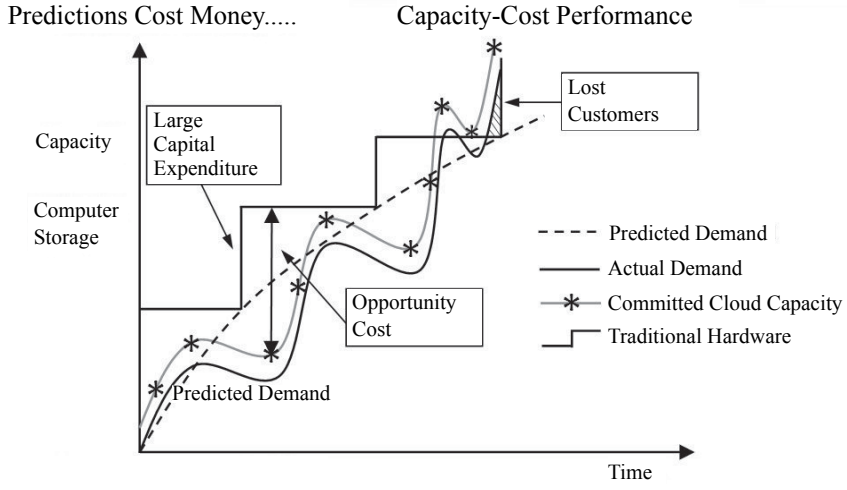
INTRODUCTION

Cloud computing has been identified as the 5th utility in the line of electricity, water, gas and 5th telephony as it enables computing to be available as and when required and paid for only what is accessed by clients. Computing resources including hardware, operating system, development tools, software applications and other services can be made available over the Internet as utilities through cloud computing (Buyya, Yeo, Venugopal, Broberg, & Brandic, 2009). Similar to any other market, the cloud computing market has also attracted a lot of service providers who host their services on the Internet for clients to access (Rimal, Choi, & Lumb, 2009). Depending on the physical resources and the loading of these systems (the number of clients accessing them), these systems will show varying service qualities (Rahim & Ku-Mahamud, 2011). Similar to any other business transaction, clients and service providers enter into a Service Level Agreement (SLA) that stipulates the conditions that must be met by both the service provider and the client. Quality of Service (QoS) would be one of the most important conditions to be met by the service provider in these agreements (Wu & Buyya, 2012). The QoS can be monitored through several attributes depending on the requirements of the customers. The dynamic nature of cloud computing requires the QoS to be monitored continuously (Patel, Ranabahu, & Sheth, 2009). Prior to entering into an SLA, customers may like to know the QoS offered by the service providers based on their preferred attribute. So a system that quantifies the QoS would be ideal for customers to identify the right service provider. Firdhous, Ghazali, Hassan, Harun, and Abas (2011) have proposed that a trust management system could be used to quantify the QoS of service providers. Several researchers have proposed trust computing mechanisms based on different algorithms and functions (Chen & Yeo, 2008; Tian, Zou, Wang, & Cheng, 2008; Yang, Qin, Wang, Liu, & Feng, 2010; Firdhous, Ghazali, & Hassan, 2011c). Almost all of these mechanisms use functions that continuously modify (increase or decrease) the trust score for every reported input value. In this paper the authors propose a novel statistically-controlled robust trust-computing mechanism, where the QoS attribute monitored is first checked whether it is within a certain confidence interval. If it is within the confidence interval, it is assumed that it has met the committed service quality and the trust score is left unchanged else the trust score is modified using a hysteresis function. Using the confidence interval to determine if the QoS attribute is within the required interval helps stabilize the fluctuations in the trust scores due to momentary variations (Firdhous et al., 2011c). Also the use of the hysteresis function to compute the trust score makes the systems more rugged in the events of malicious attacks (Morris, 2012). The combination of the above two techniques makes the proposed mechanism a robust one in the event of malicious attacks and momentary fluctuations in performance.

LITERATURE REVIEW

Cloud Computing

Distributed computing has gone through a paradigm shift in making the computing resources to clients with the arrival of cloud computing (Buyya et al., 2009). Cloud computing enables the distributed systems to make the resources available on the fly when needed and turn them off once the requirement has been satisfied. When clients outsource their computing requirements to the cloud service providers, the resource provision closely follows the resource requirements along with seasonal and daily fluctuations. The strict following of the resource provision along the resource demand helps both the service providers and the clients financially as clients can only pay for the resources consumed and the service providers can sell the same resource to more clients without affecting other clients. Figure 1 shows the Capacity-Utilization curve for computing power, which compares the fluctuating demand for computing resources and the effect of over-or-under provisioning of them (AWS, 2012). The graph also shows how cloud computing can follow the demand pattern closely against traditional hardware provisioning, where resources are either over-provisioned or under provisioned. Both over-provisioning and under-provisioning would lead to reduced economic benefits in terms of wastage of resources or lack of resources when needed most.



Source. Amazon Web Services

Figure 1. Capacity utilization curve (AWS, 2012).

The other distinction that makes cloud computing more attractive for clients to traditional outsourcing is the total absence of a commitment with regard to the resource requirements at the beginning itself (An, Lesser, Irwin & Zink, 2010). Under the traditional outsourcing model, the commitment for the resources must be made at the beginning itself and paid for the committed usage irrespective of the actual usage or demand patterns. If the actual usage is lower than the resources leased, the resources would be idle and on the other hand if the demand for the resources is higher than the resources leased, the application would suffer due to shortage of resources. From the service providers' point of view, once the resources have been committed to a client, he is unable to reallocate the resources to other clients, even if the resources are idle. Therefore with cloud computing, organizations that start small can grow big without any hindrance as the computing resources provisioning can closely follow the demand patterns. Since organizations are now required to pay only for the resources consumed, they can now invest the capital on their core business operations and other performing assets.

Cloud computing resources are hosted on virtualized platforms (Zaman & Grosu, 2010). The computer system that is to be made available as the cloud system must be first installed with a special-purpose software system called hypervisor or Virtual Machine Manager (VMM). This hypervisor creates virtual computers commensurate with CPU cycles, memory, storage and other resources on the fly when required and remove those virtual computers in a similar fashion releasing the resources once the work has been completed. On demand creation and removal of virtual computers makes it possible to market the same resource to multiple clients and help the system to follow the demand patterns closely. Selling the same hardware to multiple customers increases the utilization of the resources enabling the cost to be shared by all the customers. Sharing the resources and the cost by multiple clients results in higher utilization of the resources and lower cost for users.

Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) are the cloud services that have been commonly sold in the market (Vecchiola, Pandey, & Buyya, 2009). Under IaaS, hardware resources including CPU cycles, storage space, memory, database etc., are made available on virtualized systems. The virtualized systems thus marketed strictly resemble the real systems and can be treated as real systems for all the practical purposes. Clients can install any operating system and application as if they are installing them on real systems. Hence multiple operating systems and applications can run on a single physical computer at any time without interfering with each other. The VMM provides necessary isolation and security for these operating systems. PaaS provides complete software

development platforms on virtualized hardware systems. The development platforms thus made available include operating systems, development and testing tools, Application Programming Interface (API) etc. PaaS helps software programmers minimize the cost and time of development as they are now provided with the readymade platform instead of requiring them to purchase, install and maintain their own hardware and software (Rimal et al, 2009). SaaS is the new way of marketing software applications as a service over the Internet. Until recently, computer applications needed to be purchased outright, installed and maintained in-house (Prodan & Ostermann, 2009). Through SaaS, web-based applications developed and hosted on the cloud systems are made available as services over the Internet. Clients can access these applications over the Internet using a standard web browser. These applications include many features in common including personalization and customization to suit the specific requirements of individuals. The capability to personalize the software provides the feeling that they have been exclusively accessed by users similar to the ones installed locally. Figure 2 shows the layered architecture of cloud computing comprising the physical hardware, virtualized hardware that can be combined as cloud infrastructure layers, and IaaS, PaaS and SaaS can be combined as cloud business layers.

Communication as a Service (CaaS), Data as a Service (DaaS), Network as a Service (NaaS) and Identity and Policy Management as a Service (IPaaS) are some of the other services that are available in the cloud arena, in addition to the cloud business services described earlier (Zhou, Zhang, Zeng, & Qian, 2010). Also new services under new names are introduced to the market daily by service providers. Some researchers have combined all these services under a single name XaaS-Anything as a Service (Rao & Vijay, 2009).

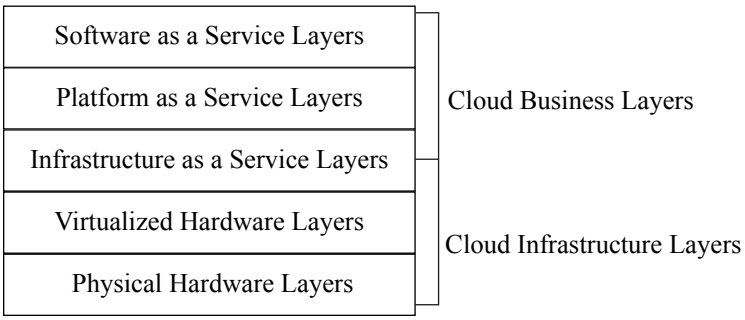


Figure 2. Layers of cloud computing (Firdhous, Ghazali & Hassan, 2011a).

Quality of Service in Cloud Computing

Cloud systems are hosted in large virtualized data centres where thousands of servers provide services to a large customer base (Garg, Gopalaiyengar, & Buyya, 2011). Though cloud services are attractive to end users in terms resource utilization and economy important issues still remain to be addressed. One of the important issues that requires immediate attention is strict QoS guarantees, so that customers will confidently outsource their computational needs to cloud services (Yeo & Buyya, 2005). Optimum resource provisioning will ensure that cloud providers maximize the utilization of their physical resources while adequately meeting their obligations to customers. Cloud data centres host a wider range of applications with different QoS requirements (Quiroz, Kim, Parashar, Gnanasambandam, & Sharma, 2009). The transactional applications demand better response times and throughput guarantees while non-interactive batch jobs are concerned with performance in terms of job-completion time and accuracy (Carrera, Steinder, Whalley, Torres, & Ayguade, 2008). Hence it can be seen that the QoS demands of the applications are more complex and depend on multiple factors or parameters.

Trust Management in Distributed Systems

Users select a suitable peer to interact with based on the trust they place on those members of the system. In distributed systems like peer-to-peer systems, grid-computing systems, cluster computing systems, sensor networks and cloud-computing systems trust-management systems are commonly employed to manage the interaction between peers (Firdhous, Hassan, & Ghazali, 2012). Several trust-computing mechanisms have been proposed in the literature based on different techniques and algorithms. Chen and Yeo have proposed a P2P trust-system based on Fuzzy decision-making (Chen & Yeo, 2008). Tian et al. (2008) have developed a trust computing model based on recommendation evidence. Yang et al. (2010) have developed a trust-computing mechanism using the entropy function. Firdhous et al. (2011c) have proposed a multilevel thresholding-based trust-computing mechanism that continuously computes and adjusts the trust values based on many confidence levels simultaneously. This mechanism helps reduce the complexity of the system as a single system can be used to serve customers with varying quality requirements. All these mechanisms employ a function that either increments or decrements the trust score continuously depending on the outcome of the most recent interaction. Firdhous et al. (2012) have proposed a hysteresis-based trust-computing mechanism for cloud computing that makes use of the memory inherent in the hysteresis function to compute a robust trust score. In this paper, the authors

further improve this mechanism by incorporating a statistical validation mechanism at the beginning of the trust-computing algorithm to reduce the impact of the momentary fluctuation of the system on the final trust scores.

STATISTICALLY-CONTROLLED TRUST-COMPUTING MECHANISM

The management of trust consists of three main functions (Carbone, Nielsen & Sassone, 2003). They are:

1. Trust Formation.
2. Trust Evolution.
3. Trust Distribution.

The first two functions, namely trust formation and trust evolution can be combined in a single unit called trust-computing unit. The initial trust score for a system that has not interacted with any client is formed using the trust-forming unit. As no previous record regarding the trustworthiness of the system exists, either a neutral value taken as the initial score or the score is computed based on system capabilities. The trust-evolution unit improves the trust scores based on the results of the interaction of the system with clients. Trust distribution unit shares the trust scores thus computed among the cooperating systems. This paper concentrates only on trust computing especially the trust-evolution mechanism. The trust distribution is left for future work.

Computing Trust

Trust has been studied by researchers working in diverse fields including sociology, psychology, economics, communication and computer science (Firdhous, Ghazali, & Hassan, 2011b). These researchers have identified several features of trust and come up with definitions based on the area of interest and perspective through which trust has been approached. The authors of this paper have adopted the following definition combining the important features of all the definitions and adapting them to suit the problem at hand.

> *"Trust is the belief on the capability of a system to carry out certain specific tasks to the satisfaction of the other, derived as a result of interaction between the systems over a period."*

The above definition focuses on the direct interaction between the systems to build trust on a system. This kind of trust is known as direct trust. If the trust is built based on recommendations from other systems that had interacted

with the system and built their own direct trust on the system, then it is known as recommendation trust (Wang, Tao, Yu, Xu, & Lü, 2005). Notationally trust can be represented as a relationship between two entities *i* and *j* as in Equation (1):

$$T: i \overset{s}{\Rightarrow} j \quad (1)$$

where, *i* provides certain services to *j* at the QoS *s* and T represents the resulting trust in *j* on *i* about its capability.

Trust-Management System

Figure 3 shows the block diagram of a trust-management system proposed in this paper. The client sends a service request to the cloud service provider along with the required performance levels and the confidence interval. The performance level is characterized by the expected or committed (in the SLA) QoS performance attribute values and the confidence level determines the stringency of the expected performance. The QoS monitor tracks the actual performance of the system and feeds the trust-computing system with the relevant QoS parameters. The trust-forming unit computes the initial trust score and feeds the trust evolution unit with it. The mid-value between the two extreme trust scores is assumed as the initial trust score in this paper for convenience. Whenever a system joins the network for the first time, it is provided with this initial neutral trust score. The trust-evolution unit receives two inputs; the client provides the system with the expected value (agreed upon in the SLA) of the QoS attribute (t_c) at the beginning of the session and the QoS monitor with the actual observed value (t_a) at the end of the session. The trust-evolution unit in detail is shown in Figure 4. The trust-evolution unit consists of the temporary buffer, confidence-interval computing module, comparator, summing point, parameter-normalization module and trust buffer. The temporary buffer stores the predetermined number of the most recent performance metric values which is input to the confidence-level computing module. The observed performance metric is compared with the confidence interval in order to determine if the observed value is within the confidence interval. If the observed value falls within the interval, the performance of the system is acceptable and the trust score is not modified. Otherwise the system computes the new trust score and modifies the stored value accordingly. The comparator acts as the switch that controls writing on the trust buffer. Figure 5 shows the trust-evolution algorithm employed in the proposed mechanism.

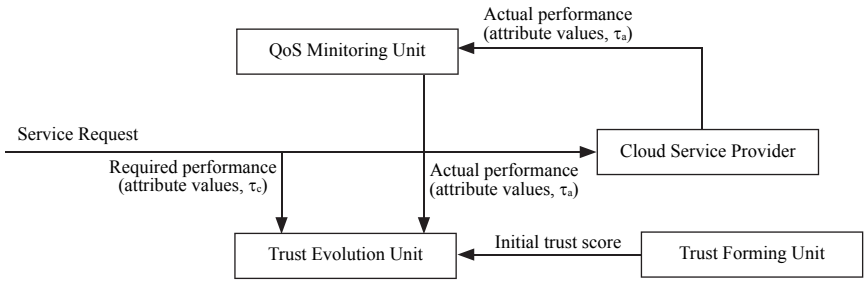


Figure 3. Trust management system.

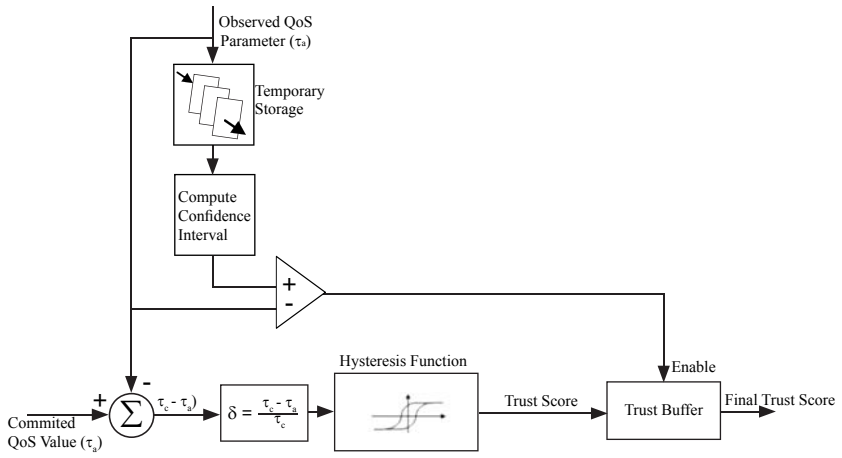


Figure 4. Trust evolution unit.

Store Actual (Observed) QoS value (τ_a) in temporary storage
Compute the confidence interval using the sample mean and sample variance Compare the QoS value τ_a with the confidence interval

if within
    No change in trust score
else

    Compute the normalized QoS parameter $\delta = \frac{\tau_c - \tau_a}{\tau_c}$
    Compute the trust score using Hysteresis function with attribute $\delta \rightarrow H(\delta)$

$$H(\delta) = \begin{cases} \frac{1-e^{-(\delta-k)}}{1+e^{-(\delta-k)}} & for\ \delta_n > \delta_{n-1} \\ \frac{1-e^{-(\delta+k)}}{1+e^{-(\delta+k)}} & for\ \delta_n < \delta_{n-1} \end{cases}$$

end if

Figure 5. Trust evolution algorithm.

Experimental Setup

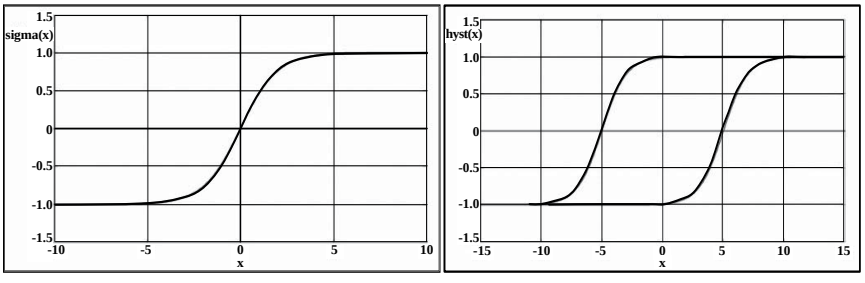
The proposed algorithm was tested for its function and accuracy with simulations. The simulation environment was set up using the GNU Octave software and the Open Source Matlab Clone. The sigmoid function defined in Equation (2) has been selected as the basis for creating a hysteresis loop required in the experiment. The shape of the sigmoid function shown in Figure 6(a). The sigmoid function has odd symmetry about both the *x* and *y* axes and asymptotically approaches for large values of *x* (Namin, Leboeuf, Muscedere, Wu, & Ahmadi, 2009).

$$sigm\,(x) = \frac{1 - e^{-x}}{1 + e^{+x}} \tag{2}$$

The hysteresis function was created by combining two horizontally shifted sigmoid functions as shown in Equation (3). The hysteresis loop thus created is shown in Figure 6 (b). The hysteresis functions have the special feature of traversing through different paths while increasing and decreasing the independent variable making the output more stable in the events of momentary fluctuations of input (Morris, 2012).

$$hyst(x) = \begin{cases} sigm\,(x-k) & for\ x_n > x_{n-1} \\ sigm\,(x+k) & for\ x_n < x_{n-1} \end{cases} \tag{3}$$

where k - amount of horizontal shift



(a) (b)

Figure 6. (a) Sigmoid function (b) Hysteresis function.

Thirty most recent sample values were stored in the temporary storage for computing the median and variance values of performance, which in turn has been used to compute the confidence interval. This number has been selected as the minimum number required to eliminate the bias inherent in

small samples (Agresti & Min, 2002). The response time of the system was selected as the QoS parameter of interest during this study for convenience. But, the selection of any QoS parameter is independent of the proposed mechanism and the mechanism would work equally well with any parameter. The required response time was fixed and the actual response times were generated using a random number generator in order to obtain consistent results. Also the randomly generated response times were trimmed to contain them within a specified range as extreme values would not be practical in real world situations.

Figure 7 and 8 show the trust values computed using statistically-validated (@ 95 and 90 per cent confidence levels) inputs and non-validated inputs against the normalized response times. The non-validated inputs were supplied to two different trust-computing mechanisms, namely hysteresis-based and entropy-based mechanisms. From these figures, it can be seen that the trust scores computed using statistically-validated inputs are more stable than that produced by non-validated inputs. In Figure 7 the trust scores computed using the validated inputs do not change at all whereas the trust scores computed using the direct inputs fluctuate heavily. The fluctuation is clearly pronounced on the entropy-based trust-scoring mechanism as the entropy functions as monotonously varying against the more stable hysteresis function. From Figure 8, it is very clear that even when the confidence level is relaxed, trust scores computed using the validated inputs are more stable changing only when the inputs fluctuate heavily but lesser than that due to the non-validated inputs irrespective of the trust-computing algorithm employed.

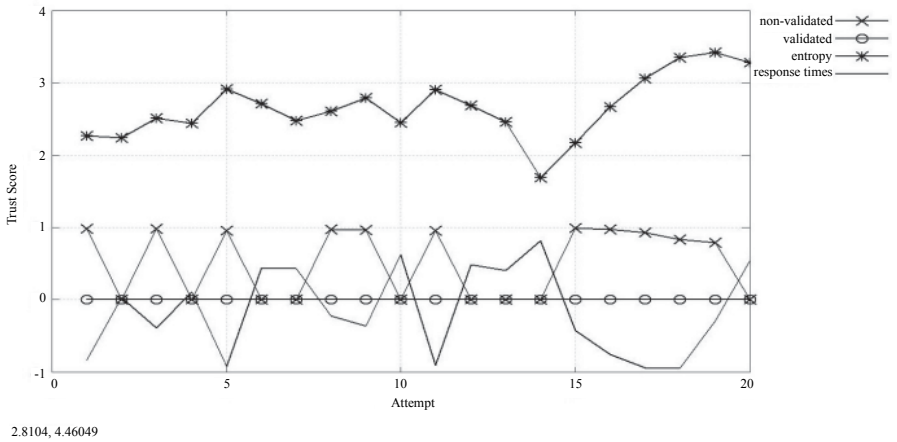


Figure 7. Trust values computed using validated @ 95 per cent and non-validated inputs.

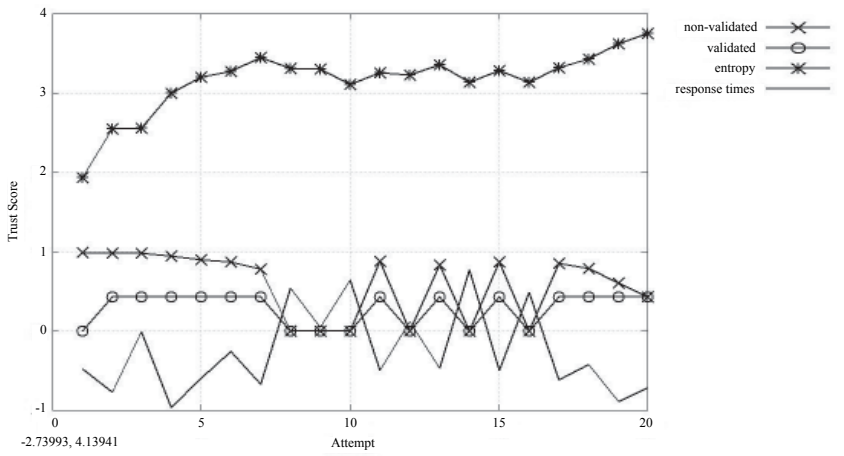


Figure 8. Trust values computed using validated @ 90 per cent and non-validated inputs.

Figure 9 shows the effect of the confidence level on the trust scores computed. When the confidence level is set at 95 per cent the fluctuation in the trust scores computed is totally eliminated compared to when the confidence level is set at 90 per cent. Hence it can be concluded that by setting a more restrictive confidence level, it is possible to eliminate the effect of fluctuations in the trust scores computed due to momentary fluctuations in performance.

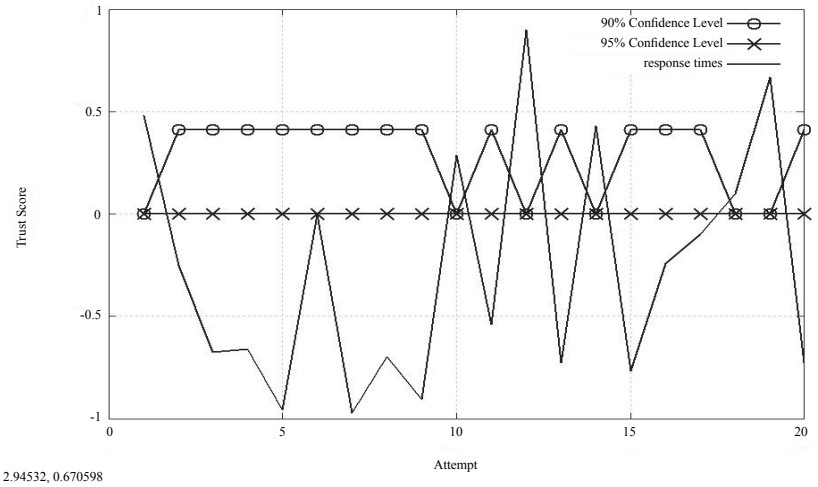


Figure 9. Effect of confidence level on the trust scores.

Figure 10 shows the change in trust scores when the deviation between the observed response time and the expected response time is large. From this figure, it can be observed that when the deviation is large both trust scores computed using the validated and the non-validated inputs using the hysteresis-based algorithm close by follow each other. Large deviations can be assumed to occur due to the actual degradation of performance rather than spurious momentary fluctuations. Hence the change in the trust scores computed using the validated inputs reflects the actual performance change in the system. It can also be observed from Figure 10 that when the hysteresis-based mechanisms follow each other closely, the entropybased mechanism shows large deviations in the computed scores. This is mainly due the fact that the entropy is a monotonously varying (increasing or decreasing) function whereas the hysteresis function stabilizes the output within a given range.

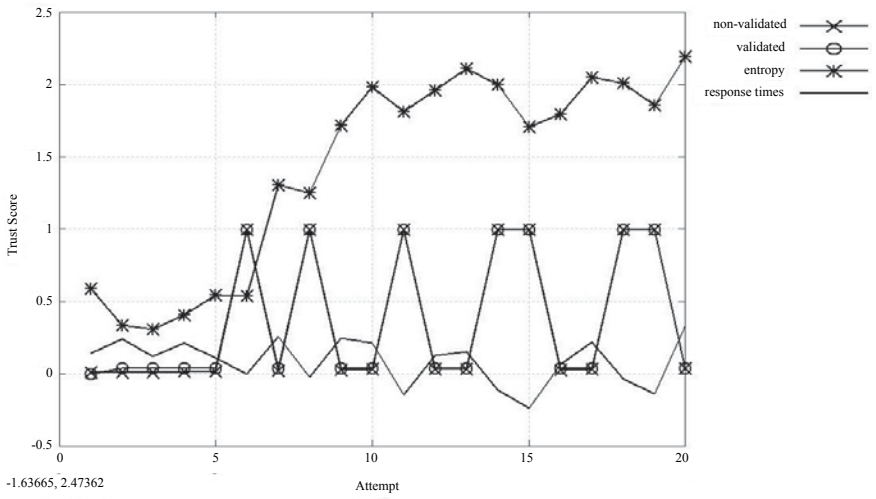


Figure 10. Effect of large deviation of input on trust scores.

Hence it can be concluded that introducing a statistical validation of the inputs to the hysteresis-based trust-computing mechanism makes it more robust by protecting the scores from fluctuations due to spurious inputs.

CONCLUSION

In this paper, the authors have presented a statistically-controlled robust trust-computing mechanism for cloud computing. Most of the trust-computing mechanisms reported in the literature employ algorithms that monotonously

change the trust scores when performance changes are reported. This is a major limitation as the trust scores thus computed would fluctuate heavily. The proposed mechanism introduces a statistical validation of the inputs with the prescribed confidence levels. This validation mechanism protects the fluctuation of the computed trust scores due to small changes in performance while letting the large deviations to go through as large deviations may actually be due to performance degradations rather than spurious fluctuations. The proposed mechanism has been tested and validated using simulations that use other mechanisms reported in the literature as reference. The results show that the proposed mechanism can perform better than other mechanisms especially the hysteresis-based trust-computing mechanism and the entropy based trust-computing mechanism in the event of momentary system fluctuations.

REFERENCES

Agresti, A., Min, Y. (2002). *On sample size guidelines for teaching inference about the binomial parameter in introductory statistics* (Unpublished). Retrieved from http://www.stat.ufl.edu

An, B., Lesser, V., Irwin, D., & Zink, M. (2010). *Automated negotiation with decommitment for dynamic resource allocation in cloud computing*. Proceedings of the 9th International Conference on Autonomous Agents and Multi Agent System pp. 981–988). Toronto, Canada.

AWS. (2012). *AWS economics center*. Retrieved from http://aws.amazon.com

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, type and reality for delivering computing as the 5th utility. *Journal of Future Generation Computer Systems, 25*(6), 599–616.

Carbone, M., Nielsen, M., & Sassone, V. (2003). *A formal model for trust in dynamic networks*. Proceedings of the 1st International Conference on Software Engineering and Formal Methods (pp. 54–61). Brisbane, Australia.

Carrera, D., Steinder, M., Whalley, I., Torres, J., & Ayguade, E. (2008). *Enabling resource sharing between transactional and batch workloads using dynamic application placement*. Proceedings of the 9th ACM/ IFIP/USENIX International Conference on Middleware (pp. 203–222). Leuven, Belgium.

Chen, H., & Yeo, Z. (2008). *Research of P2P trust based on fuzzy decision-making*. Proceedings of the 12th International Conference on Computer Supported Cooperative Work in Design (pp. 793–796). Xi'an, China.

Firdhous, M., Ghazali, O., & Hassan, S. (2011a). *A trust computing mechanism for cloud computing*. Proceedings of the Fourth ITU Kaleidoscope Academic Conference (pp. 199–205). Cape Town, South Africa.

Firdhous, M., Ghazali, O., & Hassan, S. (2011b). Trust management in cloud computing: A critical review. *International Journal on Advances in ICT for Emerging Regions (ICTer), 4*(2), 24–36.

Firdhous, M., Ghazali, O., & Hassan, S. (2011c). *A trust computing mechanism for cloud computing with multilevel thresholding*. Proceedings of the Sixth International Conference on Industrial & Information Sysems (ICIIS2011) pp. 457–461). Kandy, Sri Lanka.

Firdhous, M., Ghazali, O., Hassan, S., Harun, N. Z., & Abas, A. (2011). *Honey bee based trust management system for cloud computing*. Proceedings of the 3rd International Conference on Computing and Informatics (ICOCI 2011) (pp. 327–332). Bandung, Indonesia.

Firdhous, M., Ghazali, O., & Hassan, S. (2012). *Hysteresis-based robust trust computing mechanism for cloud computing*. Proceedings of the IEEE Region 10 Conference (TENCON 2012) (pp. 796−801). Cebu, the Philippines.

Garg, S. K., Gopalaiyengar, S. K., & Buyya, R. (2011). *SLA-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter*. Proceedings of the 11th International Conference on Algorithms and Architectures for Parallel Processing pp. 371–384). Melbourne, Australia: Springer.

Morris, K. A. (2012). What is hysteresis? *Applied Mechanics Reviews, 64*(5), 1–14.

Namin, A. H., Leboeuf, K., Muscedere, R., Wu, H., & Ahmadi, M. (2009). *Efficient hardware implementation of the hyperbolic tangent sigmoid function*. Proceedings of the IEEE International Symposium on Circuits and Systems pp. 2117− 2120). Taipei, Taiwan: IEEExplore.

Patel, P., Ranabahu, A., & Sheth, A. (2009). *Service level agreement in cloud computing*. Proceedings of the ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (pp. 1–10). Orlando, FL, USA.

Prodan, R., & Ostermann, S. (2009). *A survey and taxonomy of Infrastructure as a Service and web hosting cloud providers*. Proceedings of the 10th IEEE/ACM International Conference on Grid Computing (pp. 17–25). Banff, AL, Canada.

Quiroz, A., Kim, H., Parashar, M., Gnanasambandam, N., & Sharma, N. (2009). *Towards autonomic workload provisioning for enterprise grids and clouds*. Proceedings of the 10th IEEE/ACM International Conference on Grid Computing (pp. 50–57). Banff, AL, Canada.

Rahim, R., & Ku-Mahamud, K. R. (2011). Optimizing workload allocation in a network of heterogeneous computers. *Journal of Information and Communication Technology, 10*, 1−13.

Rao, M., & Vijay, S. (2009). *Cloud computing and the lessons from the past*. Proceedings of the 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE '09) (pp. 57−62). Groningen, The Netherlands.

Rimal, B. P., Choi, E., & Lumb, I . (2009). *A taxonomy and survey of cloud computing systems*. Proceedings of the Fifth International Joint Conference on INC, IMS and IDC (pp. 44–51). Seoul, Korea.

Tian, C. Q., Zou, S. H., Wang, W. D., & Cheng, S. D. (2008). A new trust model based on recommendation evidence for P2P networks. *Chinese Journal of Computers, 31*(2), 270–281.

Vecchiola, C., Pandey, S., & Buyya, R. (2009). *High-performance cloud computing: A view of scientific applications*. Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN) (pp. 4–16). Kaohsiung, Taiwan.

Wang, Y., Tao, Y., Yu, P., Xu, F., & Lü, J. (2005). *A trust evolution model for P2P networks*. Proceedings of the 4th International Conference on Autonomic and Trusted Computing (pp. 216−225). Hong Kong, China: Springer.

Wu, L., & Buyya, R. (2012). Service Level Agreement (SLA) in Utility Computing Systems. In V. Cardellini, E. Casalicchio, K. Castelo Branco, J. Estrella, & F. Monaco (Eds.), *Performance and dependability in service computing: Concepts, techniques and research directions* (pp. 1−25). Hershey, PA: Information Science Reference.

Yang, L., Qin, Z. G., Wang, C., Liu, Y., & Feng, C. S. (2010). *A P2P reputation model based on ant colony algorithm*. Proceedings of the International Conference on Communications, Circuits and Systems pp. 236–240). Chengdu, China.

Yeo, C. S., & Buyya, R. (2005). *Service level agreement based allocation of cluster resources: Handling penalty to enhance utility*. Proceedings of the 7th IEEE International Conference on Cluster Computing (pp. 27–30). Boston, MA, USA.

Zaman, S., & Grosu, D. (2010). *Combinatorial auction-based allocation of virtual machine instances in clouds*. Proceedings of the IEEE Second International Conference on Cloud Computing Technology and Science (pp. 127–134). Indianapolis, IN, USA.

Zhou, M., Zhang, R., Zeng, D., & Qian, W. (2010). *Services in the cloud computing era: A survey*. Proceedings of the Fourth International Universal Communication Symposium (pp. 40–46). Beijing, China.