A TRUST COMPUTING MECHANISM FOR CLOUD COMPUTING

Mohamed Firdhous[1], Osman Ghazali[2], Suhaidi Hassan[3]*

InterNetWorks Research Group, Universiti Utara Malaysia, Sintok, Kedah Darul Aman, Malaysia

Email: mfirdhous@internetworks.my[1], osman@uum.edu.my[2], suhaidi@uum.edu.my [3]

ABSTRACT

Cloud computing has been considered as the 5th utility as computing resources including computing power, storage, development platform and applications will be available as services and consumers will pay only for what consumed. This is in contrast to the current practice of outright purchase or leasing of computing resources. When the cloud computing becomes popular, there will be multiple vendor offering different services at different Quality of Services and at different prices. The customers will need a scheme to select the right service provider based on their requirements. A trust management system will match the service providers and the customers based on the requirements and offerings. In this paper, the authors propose a trust formulation and evolution mechanism that can be used to measure the performance of cloud systems. The proposed mechanism formulates trust scores for different service level requirements, hence is suitable for managing multiple service levels against single trust score. Also the proposed mechanism is an adaptive one that takes the dynamics of performance variation along with cloud attributes such as number of virtual servers into computations. Finally the proposed mechanism has been tested under a simulated environment and the results have been presented.

Keywords — Cloud Computing, Trust Formulation, Trust Evolution, Quality of Service

1. INTRODUCTION

Cloud computing has become the new paradigm in networked computing and it has been identified as the 5th utility after electricity, water, gas and telephony [1]. The emergence of cloud computing has helped organizations to change their strategy towards the investment in computing resource from own and operate to pay for what is used. For cloud computing to be accepted by a wider audience, the users need an assurance that we would receive what has been promised. This kind of assurance can be provided by a Service Level Agreement (SLA) signed between the parties. But, the clients require a method to identify the service providers who could meet their requirements. A reputation management system that quantifies the service levels would be an ideal solution from which users can select a service to suit their budgets. In this paper, the authors propose a mechanism for computing trust metrics that would form the basis for a reputation management system.

This paper is divided into six sections. Section 1 introduces the paper, while Sections 2 provides a brief introduction to cloud computing. Section 3 discusses trust and quality of service in depth and Section 4 introduces trust formulation and evolution mechanisms proposed in this paper. Simulation environment and the results are presented in Section 5. Section 6 concludes the paper along with suggestions for future work.

2. CLOUD COMPUTING

Cloud computing has been identified as the 5th utility in the line of electricity, water, telephony and gas [1]. Cloud computing has been given such a name due to the similarity between these services with respect to the way they have been accessed and paid for. Utilities have been accessed and consumed by consumers without worrying about how the services have been generated and paid only for the actual consumption of the service. With the advent of cloud computing, even the computing services will be accessed by users in a similar fashion and paid only for the services accessed. Prior to the arrival of cloud computing, computing resources were either purchased outright or leased from data center provided at fixed rates, irrespective of usage.

Cloud service providers host their services on the Internet and make them available to the prospective customers. Customers can access these services whenever they would want them and pay only for the services accessed. Service providers host their services on virtualized systems so that the same resource can be sold to multiple customers achieving maximum utilization from the resources. The virtualized systems provide the customers a sense of feeling that the resources are dedicated only for them whereas the actual resources are shared between multiple users [2]. Sharing resources this way increases the productivity of the systems while decreasing the cost of resources per user.

Cloud services are currently marketed under three different categories namely Infrastructure as Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [3].

*Mohamed Firdhous is a Senior Lecturer attached to the Faculty of Information Technology, University of Moratuwa, Sri Lanka. He is currently on leave pursuing his PhD at the Universiti Utara Malaysia.

Provision of raw computer infrastructure in terms of virtual computers is known as IaaS in cloud computing terminology. Once a virtual computer has been purchased, users can install the operating system of their choice and applications independent of other systems hosted on the same physical infrastructure. PaaS is the provision of facilities and Application Programming Interfaces (APIs) to support the complete life cycle of building and delivering web applications and services. SaaS is a model of software deployment where the user applications are hosted as a service and made available to users over the Internet [4]. Figure 1 shows the layered architecture of a typical cloud computing system. This figure includes two additional layers namely the physical hardware layer and the virtualized hardware layer in addition to the cloud service layers.

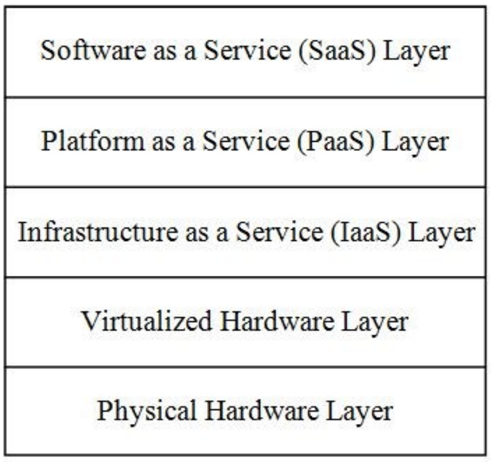
Software as a Service (SaaS) Layer
Platform as a Service (PaaS) Layer
Infrastructure as a Service (IaaS) Layer
Virtualized Hardware Layer
Physical Hardware Layer

Figure 1. Layered Architecture of Cloud System

From Figure 1, it can be seen that the cloud system is made up of five layers in total. The bottom most layer, the Physical Hardware Layer is usually made up by server class computers in data centers, clusters, grids, storage networks or any other computing systems. This is the workhorse layer which provides the necessary physical resources in terms of processors, memory, bus, storage, networking etc., to carry out the basic computing operations.

Virtualized Hardware Layer running on top of the physical hardware is created by virtualization software. The virtualization software slices the physical hardware into virtual machines in such a manner that each virtual machine will act like an independent computer running its own operating system along with other resources. These virtual computers can be pooled together to act as single resource pools. The capability of pooling the resources together makes the system elastic in the sense the virtual computers can be brought online and assigned to pools on demand. Similarly virtual computers can also be destroyed when demand subsides. This ability to create and destroy virtual computers dynamically is the basis on which IaaS is built upon. VMware, Virtual Machine Monitor (VMM), Xen, Kernel-based Virtual Machine (KVM) are some of the main products in this market.

IaaS Layer provides the clients with the facility of computing infrastructure similar to raw computing hardware [4]. Clients can install the operating system of their choice and any application development platform as if they own their own hardware. Clients are relieved from managing the physical resources such as physical computers, power and the networking but they have full control over the operating system, storage, and applications. Clients also have the flexibility of purchasing different virtual hardware components from different vendors and combine them together to form their own systems. There are several commercial IaaS providers specializing in different types of IaaS services from who customers can purchase the service they wish. Amazon provides two types of IaaS services, namely Amazon Elastic Computing Cloud (EC2) that provides flexible computing capacity and Amazon Simple Storage Service (S3) that provides flexible storage services over the Internet. IBM Smart Business Test Cloud provides a complete test environment comprising operating systems, middleware, storage, network, images and data. This reduces both the cost and time of software development drastically. The Nirvanix Storage Delivery Network and Oxygen Cloud are flexible cloud storage service that can be accessed over the internet. Interactive Intelligence provides a comprehensive set of on-demand services for cloud-based communications applications under the name of Communication as a Service (CaaS).

The Platform as a Service (PaaS) Layer extends the IaaS by abstracting it by providing an operating system and development tools creating an environment that supports the complete software development life cycle. The PaaS Layer eliminates the hassle associated with managing virtual computing instances and provides a uniform programming platform to the end user. Google’s App Engine, Amazon Elastic Beanstalk and Force.Com platform are typical PaaS offerings in the market. Google App Engine supports Python and Java programming languages along with other tools for developing and hosting web applications. The App Engine sandboxes the application to provide a secure environment for applications. The sandboxed environment isolates the application and makes it independent of the underlying hardware, operating system and physical location of the web server. App Engine also provides a distributed data storage with query and transaction processing. The Elastic Beanstalk is the PaaS service provided by the Amazon to deploy and manage any Java application in the Amazon Web Service (AWS) cloud. The Elastic Beanstalk helps any Java based web application to be loaded to the AWS as a standard Java Web Application Archive and be deployed as cloud based application. Force.com platform is a slightly different from App Engine and Elastic Beanstalk. Force.com only allows developers to create add-on application that can be integrated to the main salesforce.com application and hosted on the salesforce.com's infrastructure. These application add-ons are to be built using a proprietary Java-like programming language called Apex. The user interfaces need to be developed using Visualforce another proprietary software.

SaaS is the top most layer in the cloud services stack. Applications that were usually installed and run on individual computers are made available over the Internet as services under the SaaS. This relieves the customers from purchasing, installing, running and managing software applications. There are several commercial SaaS providers in the market and the new offerings are everyday. Google Apps, Customer Relationship Management (CRM) solution by Salesforce.com, IBM LotusLive and SAP CRM are some of the prominent SaaS offerings in the market.

Table 1 provides a summary of commercial cloud service providers along with the names and types of services offered.

Table 1. Summary of Commercial Cloud Services

Service provider	Name of service	Type of service
Amazon	Elastic Compute Cloud (EC2)	IaaS
	Amazon Simple Storage Service (s3)	IaaS
	Amazon Elastic Beanstalk	PaaS
Nirvanix	Nirvanix Storage Delivery Network	IaaS
Google	Google App Engine (GAE)	PaaS
Microsoft	Windows Azure Platform	IaaS
Rackspace	Rackspace Cloud Servers	IaaS
SalesForce	Force.com	SaaS
	Force.com Platform	PaaS
HP	HP Software-as-a-Service (Opsware)	SaaS
GoGrid	GoGrid	IaaS
ElasticHosts Ltd	ElasticStack	IaaS
Flexiant Ltd	FlexiScale	IaaS
Oracle	Sun Cloud	IaaS
IBM	Blue Cloud	IaaS

3. TRUST AND QUALITY OF SERVICE

The trust and reputation have their origin in the social sciences that study the nature and behavior of human societies [5]. Trust has been studied by researchers in diverse fields such as psychology, sociology, and economics [6]. Trust management systems play an important role in distributed systems such as peer to peer systems, grid computing, cluster computing and sensor networks [7-11]. Trust management systems help nodes to select the right peer to interact with [12].

Trust basically represents a node's competence, benevolence, integrity or predictability and any mathematical model defined to represent trust must be capable of representing all these aspects [13]. Several authors have attempted to model trust [14-17]. All these models discussed lack theoretical formulation of trust and stopped at proposing some ideas only. For Services offered in commercial would become successful only when they deliver the promised Quality of Service (QoS) [18]. A mechanism is necessary for clients to select the right service provider who could meet their requirements. A trust system built based on the QoS of different service providers will be useful in matching the capability and requirements of both service provider and clients. In this paper, the authors propose a trust mechanism based on QoS that can be used by clients to select the service providers.

QoS has been studied extensively by several researchers and reported in literature based on various QoS metrics such as response time, throughput and network utilization [18]. Xiong and Perros derive a model for computing QoS of cloud computing based on the required percentile response time [18]. They have used the M/M/1 queuing model for the analysis. Though this analysis sheds a certain amount of light into the performance of cloud computing system, the queuing model used in the analysis does not represent the real cloud environment. The cloud system is based on the virtualization of the hardware and the capability of spawning virtual machines dynamically to meet the customer requirements. Hence the model needs to be changed to M/M/n where n represents the maximum number of virtual machines that can be spawned by a physical computer in order to represent the real environment. In this paper, an analysis will be carried out based on the M/M/n queuing model for three types of customers, namely;

1. Customers who require a guaranteed level of service
2. Customers who require an average level of service
3. Customers who require basic level of service with no guarantees

Type I customers who require a guaranteed level of service would be willing to pay a comparatively large fee for the guarantee. This type of service is required for mission critical services. Type II customers who require an average level of service would pay a lower fee and would be happy when they receive the service with slight variations. This is suitable for essential but non critical services. Type III services are for non essential non real time services. The customers should be charged the lowest fee for this type of service.

4. BUILDING OF TRUST

Trust formation, evolution and propagation are central issues in trust management [13]. In this paper the authors propose a model for trust formation and evolution based the Quality of Service of cloud nodes. Figure 2 shows the proposed system that is used to form, evolve and manage the trust of computing nodes in a cloud system. The trust formulation unit computes the initial trust values based on the type of service and level of service. Service monitor monitors the performance of the service provider and informs the trust evolution unit if the service was carried out satisfactorily or not. Trust evolution unit keeps track of the current trust values for different service types and evolves the them based on the feedback received from the service monitor.

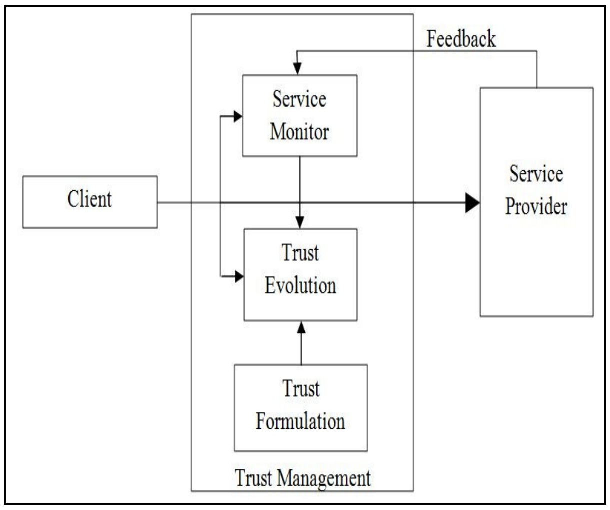

Figure 2. Trust Management System

4.1. Trust Formulation

Figure 3 shows the queuing model for the purpose of formulating trust in the cloud system. The Erlang C queuing model denoted by M/M/n in Kendall notation is used as it is the most suitable model to represent the practical cloud environment.

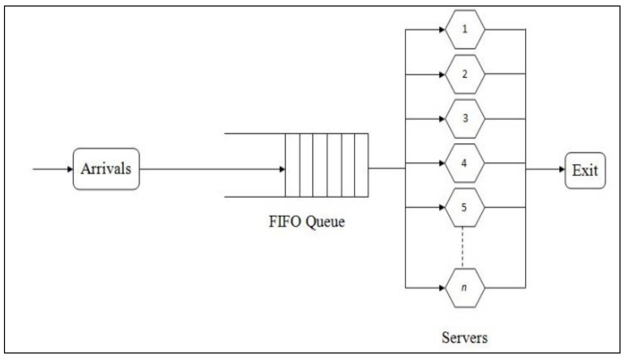

Figure 3. Queuing Model used for Formulating Trust

A FIFO queue with infinite waiting slots is assumed for simplicity. Infinite waiting slots ensure that every customer arriving at the queue be served even after a long waiting time. Every client entering the system is treated equally with no priority and treated according to the First In First Served (FIFS) discipline. Any client leaving the queue and reentering the system due to any reason is treated as a new arrival and added to the queue at the end. The arrival of requests is assumed to be Poisson distributed with a mean of λ and service time is assumed to be exponentially distributed with a mean of μ.

The initial trust values are formulated by computing the probability that the system would meet customers required response time. For example, if a customer requires the response time to be no more than τ_r and the system can meet this requirement with the lowest probability of σ. Then the initial trust score for that class of request is determined to be σ. This initial trust score will be modified according to the feedback received from customers based on the actual performance of the service provider.
Let τ, $f(t)$ and $F(t)$ represent the response time, probability distribution function and cumulative distribution respectively.

If τ_r is the required response time of the customer, the response time should satisfy eq. (1).

$$F(t)|_{t=\tau_r} = \int_0^{\tau_r} f(t)dt \geq \sigma \qquad (1)$$

where σ – the required probability

The steady state probabilities can be derived as [19];

$$p_0 = \left[\sum_{i=0}^{n-1}\frac{(n\rho)^i}{i!} + \frac{(n\rho)^n}{n!}\frac{1}{(1-\rho)}\right]^{-1}$$

and

$$p_i = \begin{cases} p_0\left(\frac{\lambda}{\mu}\right)^i \frac{1}{i!} & for\ i < n \\ p_0\left(\frac{\lambda}{\mu}\right)^i \frac{1}{n!\,n^{i-n}} & for\ i \geq n \end{cases}$$

where $\rho = \frac{\lambda}{(n\mu)}$

The probability distribution $f(t)$ of response time τ is given by;

$$f_\tau(t) = \alpha e^{\alpha t} \qquad (2)$$

where $\alpha = \left[\frac{\rho}{\lambda(1-\rho)^2}\cdot p_n + \frac{1}{\mu}\right]^{-1}$

From Eqn. (1) and (2);

$$F(t)|_{t=\tau_r} = \int_0^{\tau_r} \alpha e^{-\alpha t} dt$$

$$F(t)|_{t=\tau_r} = (1 - e^{-\alpha\tau_r}) \qquad (3)$$

Equation (3) can be used compute the initial trust score in terms QoS requirement, given the mean arrival rate, service time and the number of virtual servers.

4.2. Trust Evolution

The trust evolution module updates the trust value based on the feedback received from users. The feedback received from the users can be of two types, namely positive response where the actual response time is less than the required response time or otherwise indicating an inferior performance than required. The positive response would be used to improve the trust score while the negative response would reduce the trust score based on the algorithm shown in Figure 4.

```
required response time = $\tau_r$
actual response time = $\tau_a$
  compute normalization parameter ($\delta$) = $\frac{|\tau_r - \tau_a|}{\tau_r}$
  if ($\tau_a$ <= $\tau_r$)
    update all trust score where ($\tau_r$ >= $\tau_a$)
    $T_{n+1} = T_n + \delta * T_n$        : $T_0 = a$ and $n = 1,2...$

  else
    update all trust score where ($\tau_r$ <= $\tau_a$)
    $T_{n+1} = T_n - \delta * T_n$        : $T_0 = a$ and $n = 1,2...$

  end

          where $a$ – initial trust score computed
```

Figure 4. Trust Evolution Algorithm

This algorithm updates multiple trust scores based on the feedback received. The algorithm bases its decision on the assumption, that if the system meets a lower response time, it can meet all the response times higher than that. Also, if the system does not meet a higher response time, it cannot meet the lower response times. Initial trust score (*a*) for different response times is computed by the trust formulation unit. The normalization factor has been included into the calculation to reflect the performance of the system directly on the improvement (reduce) the trust scores.

5. SIMULATIONS

The performance of the proposed algorithm was tested using simulations. A simulation environment comprising trust formulation trust evolution, service provider and service monitor units has been setup using GNU Octave. The M/M/n queue was simulated using the qnetworks, the Queuing Networks analysis package for GNU Octave [20]. Figure 5 shows the initial trust scores computed with the mean arrival rate of 200 requests per second and a uniform service time of 75 seconds for different number of servers. From the figure, it can be seen that the initial trust scores increases drastically as the number of virtual servers increases.

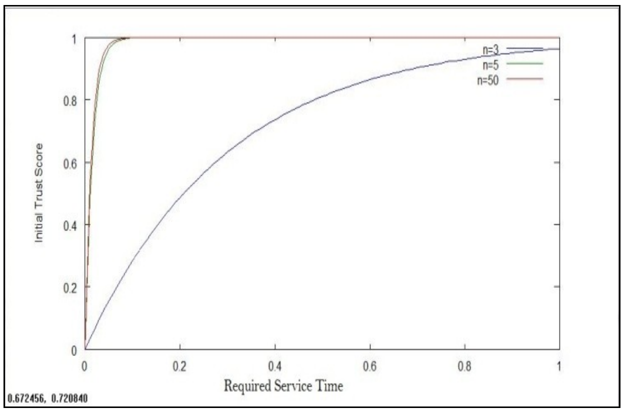

Figure 5. Initial Trust Scores for Different Service Times

Figure 6 shows the changes in trust scores due to continuous positive and negative feedbacks. The continuous negative feedbacks reduce the trust score initially at a higher rate but with time the rate also comes down though a constant time lag was used as the difference. This is due to the reason that the rate of reduction depends both on current trust value and the normalization parameter calculated using the differences in required and real response times. Similarly, during improvement of trust scores, larger trust scores responds fast to improvements compared to the smaller ones. This adaptive nature of rate of improvement (reduction) helps the system to respond to customer requirements fast as the customers who require a higher trust score would also be more sensitive to changes in trust compared to others.

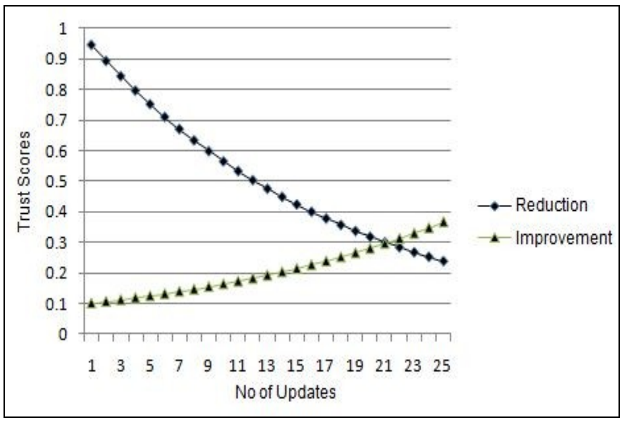

Figure 6. Change in Trust Scores

Figure 7 shows effect of response time on the trust scores. For the purpose of comparison four classes of services characterized by different response time requirements were taken into consideration. The response requirements are namely 0.1, 0.4, 0.7 and 0.9 time units. The time units have been normalized to lie between 0.0 and 1.0 for the convenience of comparison. The 0.1 is the most stringent requirement while 0.9 being the most relaxed.

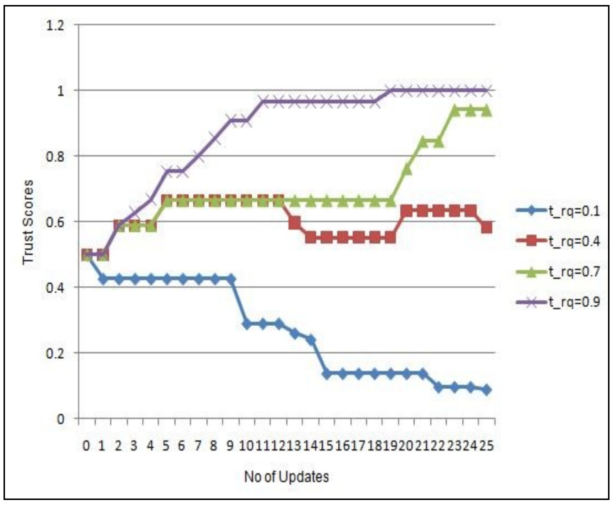

Figure 7. Comparative Change in Trust Scores

From the figure, it can be seen that whenever a more stringent requirement has been met, all the trust values of the relaxed requirements have also been improved. This is due to the reason that, if the system could meet a stringent condition it could easily meet relaxed requirements. This fact should reflect on the trust scores and hence all the respective trust values have been positively updated. Conversely, when a relaxed condition is not met, all the trust scores of the more stringent requirements are reduced. This is due to the reason that the failure to meet a relaxed requirement would necessarily an indication that more stringent performance requirements will not be met.

The figure also shows that the most stringent condition indicated by the requirement of 0.1 continues to decline. This is due to the reason that the negative performance of the system for any requirement lower than this requirement would affect this one. Hence, it is obvious from the results that it is very difficult to meet strict performance requirements unless special attention has been paid to these requirements. On the other hand, the trust score of the most relaxed performance requirement designated by the time response of 0.9 shows continuous improvement. This is due to the collective improvement of all the more stringent requirements. The other two plots show mixed results due to the random effect on their trust scores.

6. CONCLUSIONS

This paper presented a trust formulation and evolution model for cloud computing. Cloud computing has become the new paradigm in computing and accepted as the 5th utility after electricity, water, gas and telephony. For cloud computing to be accepted by different types of users, it needs to provide assurance to clients on service quality depending on the user requirements. Trust system would help users to select service providers based on the quality requirements. In this paper, the authors have proposed a trust system built based on the response time. The trust system provides a trust score between 0 and 1 for different levels of services and continues to improve these values based on the performance of the system. Hence the proposed system would be more useful for providing differentiated services at different quality levels. The proposed mechanism has been evaluated using a simulation environment setup with Octave the open source Matlab clone. The simulation results show that the proposed system works satisfactorily under constrained simulated environment. The proposed mechanism must be tested rigorously under a more open environment and in the face of adversaries in order to evaluate the ruggedness and resilience of the mechanism. The authors propose to carry out this in a future research.

REFERENCES

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 26, no. 6, pp. 599-616, June 2009.

[2] M.D de Assuncao, A. di Costanzo, and R. Buyya, "Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters," in *18th ACM international symposium on High performance distributed computing (HPDC '09)*, Munich, Germany, pp. 141-150, 2009.

[3] C. Vecchiola, S. Pandey, and R. Buyya, "High-performance cloud computing: A view of scientific applications," in *10th International Symposium on Pervasive Systems, Algorithms, and Networks*, Kaohsiung, Taiwan, pp. 4-16, 2009.

[4] R. Prodan, and S. Ostermann, "A survey and taxonomy of Infrastructure as a Service and web hosting cloud providers," in *10th IEEE/ACM International Conference on Grid Computing*, Banff, AB, Canada, pp. 17-25, 2009.

[5] H. Yu, Z. Shen, C. Miao, C. Leung, and D. Niyato, "A survey of trust and reputation management systems in wireless communications," *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1755-1772, October 2010.

[6] Z. Gan, J. He, and Q. Ding, "Trust relationship modelling in e-commerce-based social network," in *International conference on computational intelligence and security*, Beijing, China, pp. 206-210, 2009.

[7] B. Cai, Z. Li, Y. Cheng, D. Fu, and L. Cheng, "Trust decision making in structured P2P network," in *2009 International Conference on Communication Software and Networks*, Macau, China, pp. 679-683, 2009.

[8] V. Vijayakumar and R.S.D.W. Banu, "Security for resource selection in grid computing based on trust and reputation responsiveness," *International Journal of Computer Science and Network Security*, vol. 8, no. 11, pp. 107-115, November 2008.

[9] S. Mishra, D.S. Kushwaha, and A.K. Misra, "A cooperative trust management framework for load balancing in cluster based distributed systems," in *International Conference on*

Recent Trends in Information, Telecommunication and Computing, Kochi, Kerala, India, pp. 121-125, 2010.

[10] Y. Wu et al., "Automatically constructing trusted cluster computing environment," *The Journal of Supercomputing*, vol. 55, no. 1, pp. 51-68, January 2011.

[11] K.K. Tae, and S.S. Hee, "A trust model using fuzzy logic in wireless sensor network," *Journal of World Academy of Science, Engineering and Technology*, vol. 42, no. 13, pp. 63-66, 2008.

[12] M. Firdhous, O. Ghazali, S. Hassan, N.Z. Harun, and A. Abas, "Honey bee based trust management system for cloud computing," in *3rd International Conference on Computing and Informatics (ICOCI 2011)*, Bandung, Indonesia, 2011.

[13] M. Carbone, M. Nielsen, and V. Sassone, "A formal model for trust in dynamic networks," in *First International Conference on Software Engineering and Formal Methods (SEFM'03)*, Brisbane, Australia, pp. 54-61, 2003.

[14] K.M. Khan, and Q. Malluhi, "Establishing trust in cloud computing," *IT Professional,* vol. 12, no. 5, pp. 20-27, 2010.

[15] Z. Song, J. Molina, and C. Strong, "Trusted anonymous execution: a model to raise trust in cloud," in *9th International Conference on Grid and Cooperative Computing (GCC)*, Nanjing, China, pp. 133 – 138, 2010.

[16] H. Sato, A. Kanai, and S. Tanimoto, "A cloud trust model in a security aware cloud," in *10th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT)*, Seoul, South Korea, pp. 121 – 124, 2010.

[17] T.F. Wang, B.S. Ye, Y.W. Li, and Y. Yang, "Family gene based cloud trust model," in *International Conference on Educational and Network Technology (ICENT)*, Qinhuangdao, China, pp. 540 – 544, 2010.

[18] K. Xiong, and H. Perros, "Service performance and analysis in cloud computing," in *World Conference on Services - I*, Los Angeles, CA, USA, pp. 693 – 700, 2009.

[19] N.J. Gunther, *Analyzing computer system performance with Perl:PDQ*. Berlin Heidelberg, Germany: Springer-Verlag, 2005.

[20] M. Marzolla, "The qnetworks toolbox: a software package for queueing networks analysis," in *17th International Conference on Analytical and Stochastic Modeling Techniques and Applications (ASMTA 2010)*, Cardiff, UK, pp. 102-116, 2010.