International Journal of Database Theory and Application
Vol.7, No.4 (2014), pp.1-10
http://dx.doi.org/10.14257/ijdta.2014.7.4.01

Market Mechanism Based Resource Management Strategy in Cloud Computing

Lina Ni[1] and Jinquan Zhang[2]

College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao, China
[1]nln2004@163.com, [2]tjzhangjinquan@126.com

Abstract

Cloud computing is an emerging web-based commercial computing paradigm in which the dynamic optimization allocation of resources is the core issue. This paper proposes a market mechanism model of resources binding service dynamically. First, the resources usability evaluation strategy based on D-S theory is given. Then we present fair value based resource pricing strategy and resource dynamic optimization allocation algorithm to achieve efficient allocation of system resources and long-run equilibrium. Simulation results show that the proposed model and strategy are effective in managing the cloud computing resources.

Keywords: *Resource Management, Cloud Computing, D-S theory, Availability Evaluation*

1. Introduction

Cloud computing is a new and promising commercial computing paradigm which is developed from distributed computing, parallel computing and Grid computing. It means that Internet computing and storage have considerably evolved from small isolated nodes to large-scale Cluster-like architectures driven by efficiency and scalability needs [1, 2]. Cloud computing is a service model for IT provisioning, often based on virtualization and distributed computing technologies. Within the cloud paradigm, concepts such as virtualization, distributed computing and utility computing are applied [3]. Computing resources are offered over the Internet as scalable, on-demand Web services. Cloud services are available at different layers:

- *SaaS*: The data Storage as a Service delivering basic storage capability over the network;
- *IaaS*: The Infrastructure as a Service layer providing bare virtual hardware with no software stack;
- *PaaS*: The Platform as a Service layer providing a virtualized servers, OS, and applications;
- *SaaS*: The Software as a Service layer providing access to software over the Internet as a service.

Resource management is the main task of cloud computing. Compared with grid computing, the focus of its resource management problem is transformed to resource virtualization and allocation rather than job decomposition and scheduling. It is more urgent to find method of allocating resources reasonably according to the complexity of user's requests and the cost they're willing to pay. That renders the Internet a large repository where resources are available to everyone as services the resources are processor time or storage space, while the economic actors are computers or web services [4]. The complex interdependencies are broken down into

ISSN: 2005-4270 IJDTA

two types of interrelated markets [5]: (1) a service market- which involves trading of application services, and (2) a resource market- which involves trading of computational and data resources, such as processors, memory, virtual machines or storage space. The service market simulates a Software-as-a-Service (SaaS) delivery model [6], which composes basic services dynamically as needed. The complex service's functionality is configured and bound at delivery time to a user of the application layer network. Commercial platforms like Sales force or Google's Platform-as-a-Service (PaaS) offerings exemplify the practical importance of the service market model.

The resource market trades resource bundles, which could be interpreted as virtual machines of different size and capacity. The Infrastructure-as-a-Service (IaaS) model describes the delivery of virtualized resource bundles to applications (basic services). Reference [7] proposes the DIET-Solve Grid middleware on top of the EUCALYPTUS cloud system to demonstrate general purpose computing using cloud platforms. In [8], a new framework is presented that provides efficient green enhancements within a scalable cloud computing architecture. Using power-aware scheduling techniques, variable resource management, live migration, and a minimal virtual machine design, overall system efficiency will be vastly improved in a data center based cloud with minimal performance overhead. The architecture for market-oriented allocation of resources within clouds provides thoughts on market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain SLA-oriented resource allocation [9].

The remainder of this paper is organized as follows. Section 2 presents the resource availability evaluation strategy based on D-S evidence theory. Then the fair value based resource pricing strategy and resources dynamic optimization allocation algorithm are proposed in Section 3. Section 4 demonstrates the simulation experiment to validate the algorithm. Finally conclusion and future work are given in section 5.

2. Resource Availability Evaluation Strategy Based on D-S Evidence Theory

2.1. Principle of the Strategy

Suppose that all the services be relatively independent. When a service considers the resource availability, all the other services interacted with this resource become the recommenders of it and the evidences provided are the recommended results.

Let Q be a question, all its possible answers can be denoted as set U . In this paper, it is supposed that $U = \{Su, \neg Su\}$, where Su represents a successful service of a resource and $\neg Su$ represents the unsuccessfulness of a resource in this service, respectively. Assume that $s \in \{Su, \neg Su\}$ be the right answer of question Q . We expect to obtain the right answer s in virtue of the recommendation.

The resource availability evaluation only according to the experience of the past usage may not completely objective. Because of the incompleteness, imprecision and incomplete reliability of evidences (for example, there may exist a newly joined resource, or a record without interactive behavior with services, and so on), we are unable to determine s . However, determining the scope of s in some degree based on the recommended evidence is possible.

Fortunately, evidence theory can treat the evaluation provided by each service providers as recommended evidence and thus calculate the resource availability.

The evidence theory, as introduced by Dempster and extended later by Shafer, is concerned with the question of belief in a proposition and systems of propositions, which is also called D-S theory [10, 11]. D-S theory introduces belief function and satisfies probability weak axiom, which allows the representation of both imprecision and uncertainty. Rather than computing prior probabilities and conditional probabilities of propositions, it computes probabilities that

evidence supports the propositions and offers an alternative approach to dealing with uncertainty reasoning based on incomplete information. D-S evidence theory has been widely used in the imprecise inferring in expert systems [12]. Despite many mathematical models based on D-S evidence theory currently exist in different family of applications, there is scant literature for web service and resource management. Zhang [13, 14] proposes task seamless active transfer algorithm via applying evidence theory to pervasive computing.

We'll realize resources availability evaluation with the aid of assignment function, belief function and plausibility function of D-S evidence theory in the following.

2.2. Basic Notions

Definition 1. Let U be a nonempty finite set, function $m : 2^U \to [0,1]$, such that

(1) $m(\phi) = 0$, where ϕ denotes an empty set and

(2) $\Sigma_{A \in 2^U} m(A) = 1$,

then m is called *Basic Probability Assignment (BPA) function* on the power set of U. $m(A)$ is called the *Basic Probability Assignment (BPA)* of A.

In this paper, m signifies the degree of QoS guaranteed by a resource or the degree of ignorance of this resource. Each service generates a *BPA function* m for resources. Thus,

$m(\{Su\}) + m(\{\neg Su\}) + m(\{Su, \neg Su\}) = 1$.

The *BPA of* U 's all subsets: $\{Su\}$, $\{\neg Su\}$ and $\{Su, \neg Su\}$ can be obtained as follows. We give the definition of *Basic Availability Evaluation (BAE).*

Definition 2. A service's *Basic Availability Evaluation (BAE)* bae_{ij}^k in each time service S_j interact with resource r_i is defined by the following expression:

$bae_{ij}^k = \alpha_1 respond_i + \alpha_2 renum_i + \alpha_3 succ_i$, $\alpha_1 + \alpha_2 + \alpha_3 = 1$,

where $respond_i$, $renum_i$ and $succ_i$ represent the indexes corresponding to a service of response time, reboot times and successful states (i.e., successfulness, unsuccessfulness), respectively. $\alpha_i (i = 1,2,3)$ represents the weight of each index, that is, the degree of a service caring for the above indexes.

The specific method to determine these indexes are elaborated in the following:

- $respond_i$

Generally, the service should be finished in the agreed time. If the user's requirement to the service's response time is not very harsh, but allows the service's response time exceed the agreed time, the service price can be adjusted appropriately. τ_1 denotes the agreed response time, and c is the price that a user paid if a service finished in τ_1. τ_2 represents the longest response time that a user can endure. Then, the response time τ is among τ_1 and τ_2, the user paid price $c' = c \times (\tau_2 - \tau)/(\tau_2 - \tau_1)$. Thus, the response time of the resource serviced $respond_i$ is defined as:

$$respond_i = \begin{cases} 1 & \tau \le \tau_1 \\ (\tau_2 - \tau)/(\tau_2 - \tau_1) & \tau_1 \le \tau \le \tau_2 \end{cases}.$$

- $renum_i$

$renum_i$ scales the resource service stability. Obviously, the less the numbers of the service's reboot, the higher the stability of the resources, and vice versa. Suppose that the maximum number of reboot that a user and resources (services) provider agreed is $n_{\max}$. The number of reboot in actual service is $n_{fac} (0 \le n_{fac} \le n_{\max})$. Then, we have:

$renum_i = 1 - n_{fac}/n_{\max}$.

- $succ_i$

$succ_i$ scales whether a service is successful or not. It is related to the service's response time $respond_i$ and reboot number $renum_i$. $succ_i$ is defined as follows:

$$succ_i = \begin{cases} 0 & \tau \geq \tau_2 \quad or \quad n_{fac} \geq n_{\max} \\ 1 & others \end{cases}.$$

Obviously, the closer the service from now can more reflect the state of the resources. Therefore, the *Basic Availability Evaluation (BAE)* of service s_j to resource r_i is defined as:

$$bae_{ij} = \Sigma_{k=1}^{n} \beta_k bae_{ij}^k \ , \quad \Sigma_{k=1}^{n} \beta_k = 1$$

where β_k is the weight defined by the user. $k = 1$ represents that the service is the closest from now and the remaining is on the analogy of this. Generally, $\beta_k \geq \beta_p \ \ (k < p)$.

Suppose that the unknown situations of service s_j for resources r_i is $unk_{ij} (0 \leq unk_{ij} \leq 1)$. Then, we define:

$$m(\{Su\}) = bae_{ij}(1 - unk_{ij}) \ ,$$

$$m(\{\neg Su\}) = (1 - bae_{ij})(1 - unk_{ij}) \ ,$$

$$m(\{Su, \neg Su\}) = unk_{ij} \ .$$

Definition 3. Bel is said to be *belief function* on the power set of U if m is the *BPA* function on the power set of U, function $Bel: 2^U \to [0,1]$ satisfies

$$\forall A \in 2^U, Bel(A) = \Sigma_{B \subset A} m(B) \ .$$

It can be seen from Definition 3 that $Bel(\phi) = m(\phi) = 0$ and $Bel(U) = 1$. The *belief function* and the *BPA* function may determine mutually, that is

$$Bel(\{Su, \neg Su\}) = m(\{Su\}) + m(\{\neg Su\}) + m(\{Su, \neg Su\}) = 1 \ .$$

Therefore, they are the different denotations for the same evidence. It is not difficult to obtain the following inequality from Definition 3:

$$Bel(A) + Bel(\sim A) \leq 1 \ ,$$

where $\sim A$ denotes the complementary set of *A* to U. This is one of the basic distinctions between *belief function* and *BPA function*.

The *belief function* value $Bel(A)$ can be viewed as the total support degree of recommended evidences to a proposition or the degree of a service believes in the proposition in this evidence. It is also interpreted as the minimum uncertainty value of *A*.

Definition 4. Let two recommended evidences on U be completely independent, their *BPA functions* on the power set of U be m_1 and m_2, respectively. Defining $m = m_1 \oplus m_2$ is *Combination BPA function* of the two evidences to proposition, which carries the joint information from the two evidences. The evidence *combination BPA function* satisfies the following:

(1) $m(\phi) = 0$,

(2) $m(A) = k\Sigma_{X \cap Y = A} m_1(X) m_2(Y)$,

where $k^{-1} = \begin{cases} 1 - \Sigma m_1(X) m_2(Y) & X \cap Y = \phi \\ \Sigma m_1(X) m_2(Y) & X \cap Y \neq \phi \end{cases}$.

If $k^{-1} = 0$, then there does not exist *Combination BPA function*, which indicates the two evidences are contradictory. m reflects the combinational degree of recommended evidences corresponding to m_1 and m_2 to a proposition. k is a normalized factor, which assures the range of m is [0,1] and is often interpreted as a measure of conflict between the evidences. The larger

the value of k is, the more conflicting are the evidences, and the less informative is their combination.

2.3. Resource Availability Computing

The evidences of a service evaluating the resources come from two parts: one part is obtained from the service itself interacted with the resources; another part is the evidence recommended from other service. In general, a service will much more believe the evidence obtained itself, because the evidences recommended by other services may be malicious or intentionally enhance the resources' *BPA* for the sake of obtaining higher resources utilization ratio. We distinguish these evidences through the weighting way and punish the malicious evidences by reducing their weighted value.

We suppose that there exist *n* services $s_1, s_2, \cdots, s_n$, which corresponding weighted values are $\omega_1, \omega_2, \cdots, \omega_n$ such that $\Sigma_{i=1}^{n} \omega_i = 1$. We also suppose that there be k resources $r_1, r_2, \cdots, r_k$. The *BPAs*: $m(\{\neg Su\})$ and $m(\{\neg Su\})$ of service s_i for resource r_j are α_{i1} and α_{i2}, such that $\alpha_{i1} + \alpha_{i2} \le 1$, respectively. The support degree of each service s_i to resource r_j is defined as follows:

$$m_{ij}(\{Su\}) = \omega_i \alpha_{ij1}, \quad m_{ij}(\{\neg Su\}) = \omega_i \alpha_{ij2},$$

$$m_{ij}(\{Su, \neg Su\}) = 1 - (m_{ij}(\{Su\}) + m_{ij}(\{\neg Su\})).$$

Since the evidence submitted by each service to the resources is independent and according to Definition 3, the *Combination BPAs* of service s_i and s_p to resource r_j:

$$m_{i \oplus p,j}(\{Su\}) = k_{ip}(m_{ij}(\{Su\})m_{kj}(\{Su\}) + m_{ij}(\{Su, \neg Su\})m_{kj}(\{Su\}) + m_{ij}(\{Su\})m_{kj}(\{Su, \neg Su\}))$$

$$m_{i \oplus p,j}(\{\neg Su\}) = k_{ip}(m_{ij}(\{\neg Su\})m_{kj}(\{\neg Su\}) + m_{ij}(\{Su, \neg Su\})m_{kj}(\{\neg Su\}) + m_{ij}(\{\neg Su\})m_{kj}(\{Su, \neg Su\}))$$

$$m_{i \oplus p,j}(\{Su, \neg Su\}) = k_{ip} m_{ij}(\{Su, \neg Su\})m_{kj}(\{Su, \neg Su\}),$$

$$1/k_{ip} = 1 - (m_{ij}(\{Su\})m_{kj}(\{\neg Su\}) + m_{ij}(\{\neg Su\})m_{kj}(\{Su\})).$$

Thus, the *Combination BPA* of *n* services to one resource can be obtained through combining the first *n-1* services' *Combination BPA* with the last service's *BPA*. Therefore, *n* services' *Combination BPA* may be solved through recursive method. Assuming that m_j is the *Combination BPA* of all services to resource r_j. Thus, function m_j reflects the availability evaluation of every service to resource r_j. For two alternative resources r_1 and r_2 of service s_i, their corresponding *BPA functions are* m_1 and m_2. If $m_1(\{Su\}) \ge m_2(\{Su\})$ and $m_1(\{\neg Su\}) \le m_2(\{\neg Su\})$, service s_i will select resource r_1 naturally because the availability of r_1 is better than that of r_2 obviously. However, if $m_1(\{Su\}) \ge m_2(\{Su\})$ and $m_1(\{\neg Su\}) \ge m_2(\{\neg Su\})$, it is not simply considered that the availability of r_1 is better than that of r_2. One feasible method is carrying on weighting processing, i.e., the resource availability can be defined as:

$$utl(r_j) = \lambda_1 m_j(\{Su\}) - \lambda_2 m_j(\{\neg Su\}) + \lambda_3 m_j(\{Su, \neg Su\}),$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

3. Fair Value Based Resource Pricing and Allocation Strategies

The fair value of resource is that a mutually acceptable price of a kind of resource by resource provider and resource consumer (hereinafter referred to as the user) in the long-term use will remain unchanged in the next period of time. That is to say, resource providers as long

as are willing to provide services who must accept this price. Similarly, the user will also have to accept resource at this price, regardless of whether the QoS guarantee. Obviously, if a resource is always cannot guarantee QoS, there would no user like to use this resource after a period of time. Thus, computing the resource's fair value would have a lower resource price evaluation in the later.

The user decides whether to choose this resource according to the former evaluation of other users before each time use it. Generally, resources with higher availability may provide service for the user with higher credibility in the service binding resources dynamically. Meanwhile, considering the higher credibility user may join in the system dynamically, the algorithm will reserve partial resources with higher availability for them. Therefore, we define the two thresholds η_1 and η_2 for user's credibility [15] and resource availability, respectively. If user u_j is with credibility $up(u_i) \geq \eta_1$, he can use resource r_j with availability $utl(r_j) \geq \eta_2$, also can use other resources, and vice is not.

Assume that there exists n services $s_1, s_2, \cdots, s_{n_i}$, and that service s_i can utilize q_i resources $r_{i1}, r_{i2}, \cdots, r_{iq_i}$. Then, the procedure of service selection algorithm is given as follows:

Algorithm ***Selection Algorithm of Service Binding Resource***

Step 1 Suppose that service s_i plans to map to users $u_{i1}, u_{i2}, \cdots, u_{ip_i}$;

Step 1.1 Compute the credibility $up(u_{i1}), up(u_{i2}), \cdots, up(u_{ip_i})$ for user $u_{i1}, u_{i2}, \cdots, u_{ip_i}$. Sort the credibility as $up(u_{i1}'), up(u_{i2}'), \cdots, up(u_{ip_i}')$ in decreasing order.

Step 1.2 Suppose that the credibility of d users such that

$$up(u_{i1}') \geq \eta_1, up(u_{i2}') \geq \eta_1, \cdots, up(u_{id}') \geq \eta_1 \quad (d \leq p_i),$$

denote $uh = [u_{i1}', u_{i2}', \cdots, u_{id}']$, $ul = [u_{i,d+1}', u_{i2}', \cdots, u_{ip_i}']$;

Step 2 Suppose that service s_i can utilize q_i resources $r_{i1}, r_{i2}, \cdots, r_{iq_i}$;

Step 2.1 Compute the availability $utl(r_{i1}'), utl(r_{i2}'), \cdots, utl(r_{iq_i}')$; Sort the availability as $utl(r_{i1}'), utl(r_{i2}'), \cdots, utl(r_{iq_i}')$ in decreasing order;

Step 2.2 Suppose that the availability of c resources such that

$utl(r_{i1}') \geq \eta_2, utl(r_{i2}') \geq \eta_2, \cdots, utl(r_{ic}') \geq \eta_2$, $(c \leq q_i)$,

denote $rh = [r_{i1}', r_{i2}', \cdots, r_{ic}']$, $rl = [r_{i,c+1}', r_{i2}', \cdots, r_{iq_i}']$;

Step 3 Suppose that for any resource in $r_{i1}', r_{i2}', \cdots, r_{iq_i}'$, the number of users which it can admit be $e_t (1 \leq t \leq q_i)$;

Step 4 Fetch resource r_{ie_t}' from rh in turn;

Step 4.1 If $e_t < d$, then

bind service s_i with resource r_{ie_t}',

map them to the first e_t user registries in uh,

delete the users from uh, that is, $d = d - e_t$;

Step 4.2 Fetch next resource from rh and repeat *Step 4.1* until either uh or rh is empty;

Step 5 If uh is not empty, then put the users in uh in the front of ul in turn. Also denote by ul;

Step 6 Schedule resources in rl and users in ul in the manner similar to *Step 4*. □

4. Simulation Experiment

In this experiment, we simulate the allocation of computing resources to verify the performance of the proposed strategy. The simulated cloud computing environment is constituted by a management node and four computing nodes. Management node is responsible for receiving user tasks and assigning them to the computing nodes. Computing nodes execute the assigned tasks according to their own scheduling algorithm.

On each computing node, two functions f_1 and f_2 are established. Here f_1 simulates whether a task needs to be rebooted. When a task needs to be rebooted, the computing node needs to obtain the relevant information from the management node again. f_2 simulates whether a task can successfully complete. When the value of f_2 is true, the completion time of the task will return to the management node. Similarly, function f_3 is established for each user to simulate whether a user completely uses this service. That is, when the value of f_3 is true, the user will ensure that he can wait for the service results, otherwise, the value is false. In this way, the computing node can choose whether to accept the assigned tasks from the management node. That is to say, if the service is not being accepted, the management node needs to choose computing node again.

In the simulation experiment, we generate a set of services stochastically and select the computing node employing the above mentioned algorithm. Figure 1 gives the availability evaluation of five types of services in the management node to one computing node after 10 times usage. If the service does not use this computing resource, this availability evaluation value will keep the original value.

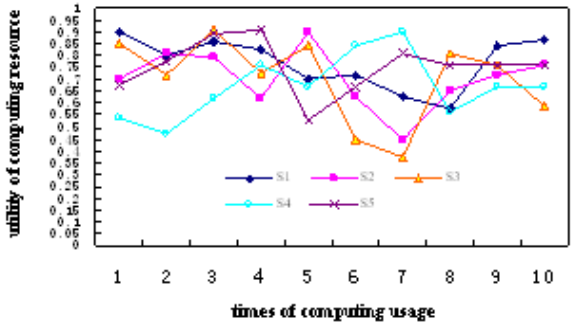

Figure 1. Availability Evaluation of Five Types of Services to Resources

Figure 2 gives the circumstances of successful service under the condition that the ratio of unsuccessful services is 10% when $\eta_1 = 0.8$ and $\eta_2 = 0.9,0.8,0.7,0.6,0.5,0.4$. As seen in Figure 2, with the decrease of η_2, the ratio of successful service presents the drop tendency. This is mainly because the service may bind some resources with lower availability and cannot be successfully served.

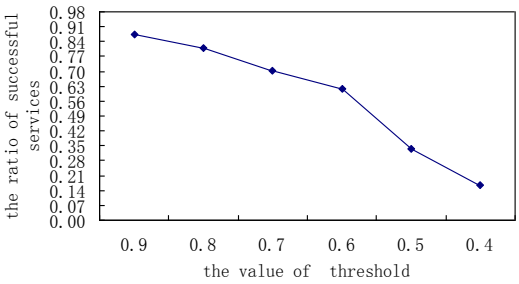

Figure 2. The Ratio of Successful Services under Different Ratio of η_2

Figure 3 presents the circumstances of successful service under the condition that the ratio of unsuccessful services is 10% when threshold $\eta_1 = 0.9, 0.8, 0.7, 0.6, 0.5, 0.4$ and $\eta_2 = 0.8$, respectively. Similarly, with the decrease of η_1, the ratio of successful service changes irregularly. The reason is that the service used by a user with lower credibility may bind some resources with lower availability or with higher availability, which leads to the uncertainty of the successfulness of the services.

Figure 4 illustrates the comparison between the service selection strategy without considering the user's credibility and resource's availability and our selection algorithm. The

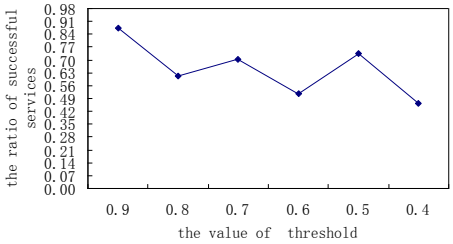

Figure 3. The Ratio of Successful Services Under Different Ratio of η_1

random service selection strategy may select resources stochastically to bind with services. Our algorithm is of high successful ratio, and thus the service efficiency is enhanced in our model.

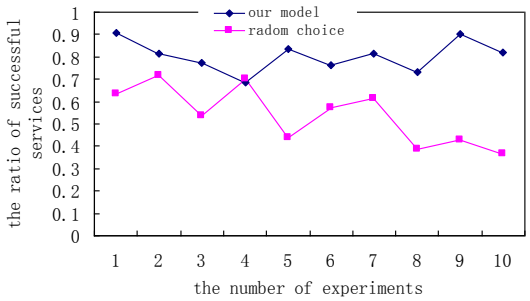

Figure 4. Comparison of a Random Allocation and Our Service Availability Guided Allocation for Selecting Successful Services in 10 Experiments

5. Conclusion

Cloud computing is an emerging commercial computing paradigm in which the resources management is the core issue. Resources may join in or withdraw from the system. The QoS of new joined resources is uncertain for service. Aiming at this uncertainty, this paper proposes a new resource availability evaluation method, and applies it into service binding resources. The further work is to research the resource configuration optimization dynamically, improve resource allocation efficiency.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under grant 61170078, the National Basic Research Program of China under Grant 2010CB328101, Doctoral Fund of Ministry of Education of China under Grant 20133718110014, 20113718110004, Natural Science Foundation of Shandong Province of China under Grant No. ZR2012FM003. This work is also supported by Project of Shandong Province Higher Educational Science and Technology Program under Grant J13LN18, Qingdao Science Plan Application Basic Research Project under Grant 12-1-4-6-(9)-jch, 13-1-4-116-jch, and the SDUST Research Fund of China under grant 2011KYTD102.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph *et al*., "Above the clouds: a Berkeley view on cloud computing," Technical Report UCB/EECS-2009-28, Electrical Engineering and Computer Sciences, University of California at Berkeley, February **(2009).**
[2] L. M. Vaquero, L. R. Merino, J. Caceres, and M. Linder, "A break in the clouds: towards a cloud definition," ACM SIGCOMM Computer Communication Review, vol. 39, January **(2009),** pp.50-55.
[3] A. Lenk, M. Klems, J. Nimis, S. Tai *et al*., "What's inside the cloud? an architectural map of the cloud landscape," Proceedings of IEEE Symptom on Software Engineering Challenges of Cloud Computing(ICSE 09), Vancouver, Canada, May **(2009),** pp. 23-31.
[4] R. Buyya, H. Stockinger, J. Ghiddy and D. Abramson, "Economic models for management of resources in peer-to-peer and grid computing," Concurrency and Computation: Practice and Experience, vol. 14, November **(2002)**, pp.1507-1542.
[5] W. Streitberger and T. Eymann, "A simulation of an economic, self-organising resource allocation approach for application layer networks," Computer Networks, vol. 53, July (**2009),** pp.1760-1770.
[6] M. Turner, D. Budgen and P. Brereton, "Turning software into a service," IEEE Computer, vol.36, October **(2003),** pp. 38-44.
[7] C. Eddy, D. Frederic and L. David, "Cloud computing resource management through a Grid middleware: A case study with DIET and eucalyptus," Proceedings of IEEE Symptom on Cloud Computing (CLOUD 09), Vancouver, Canada, May **(2009).**
[8] A. J. Younge, G. V. Laszewski, W. Lizhe, S. Lopez-Alarcon *et al*., "Efficient resource management for Cloud computing environments", Proceedings of IEEE Symptom on Green Computing, Chicago, IL, USA, August **(2010),** pp. 357-364
[9] Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented Cloud computing: vision, hype, and reality R for delivering IT services as computing utilities," Proceedings of the 10th IEEE Symptom on High Performance Computing and Communications, Dalian, china, September **(HPCC 2008),** pp**.** 5-13.
[10] G. Shafer, "A Mathematical Theory of Evidence", Princeton University Press, Princeton, NJ, **(1976)**.
[11] K. Sentz and S. Ferson, "Combination of Evidence in Dempster-Shafer Theory", Sandia Report, April **(2002).**
[12] W. S. Wang, "The Principles and Application of Artificial Intelligence", Electronic Industries Publishing Company, September **(2000).**
[13] D. G. Zhang, G. Y. Xu, Y. C. Shi, H. Zhao *et al*., "Extended Method Of Evidence Theory For Pervasive Computing", Journal of Computer, vol.27, iss. 7, **(2004),** pp. 918-927.
[14] D. G. Zhang, Y. C. Shi, G. Y. Xu and E. Y. Chen, "Algorithm of Task-oriented Seamless Active Transfer for Pervasive Computing", Journal of Computer, vol.28, iss. 6, **(2005)**, pp. 818-831.
[15] L. N. Ni, "Modeling and analyzing of Grid resources management system based on generalized Petri nets," Doctoral Dissertation, **(2009)**.

Authors

Lina Ni, she received her Ph.D. degree in Computer Software and Theory from Tongji University, China, in 2009. Now she is an associate professor of the College of Information Science and Engineering, Shandong University of Science and Technology. Now she is a senior member of CCF. Her current research interests include cloud computing, service computing, big data, intelligent computing, Petri net theory and application.

Zhang Jinquan, he received his Ph.D. degree in Computer Application Technology from Tongji University, China, in 2007. Now he is an associate professor of the College of Information Science and Engineering, Shandong University of Science and Technology. His current research interests include Petri net theory and application, distributed computing, and cloud computing.