VOL. 10, NO. 5, MARCH 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

CLOUD BASED SEARCH ENGINE

Nithya G.[1], Engels M. S.[2], Gayathri S.[1] and Ganesh Kumar D.[1]
[1]Department of Information Technology, Sri Krishna College of Engineering and Technology, Tamil Nadu, India
[2]Department of Computer Science and Engineering, PSG College of Technology, Tamil Nadu, India
E-Mail: nithyag@skcet.ac.in

ABSTRACT

With the advancement in science and technology, the problem of managing and maintaining expensive computing resources have become more complicated. In order to overcome this burden the recent trend is to effectively use cloud computing, supporting resource sharing, with many services. The goal is to aggregate idle network and to preserve resources such as CPU cycles and storage spaces to facilitate effective utilization and computing. For the above mentions, we need to find an efficient method for identifying the services based on all the results from a cloud based search engine. The assignment and choosing of a cloud service should facilitate efficient problem solving and promote optimal use of resources. One such solution must be able to apply in a large array of information processing units. This project proposes to address the above problem using the Multi-Agent Brokering Approach, for the identification of services from the results of a cloud based search engine in a cloud environment. The Multi-Agent Approach ensures that the agents can specialize in identification of services made to process the requests. It also orders the results of the search with various ordering options such as CPU speed, memory and storage. For achieving this, the proper representation of provider capabilities and ontology relationships are essential.

Keywords: cloud computing, ontology, multi agent, scheduling.

1. INTRODUCTION

Cloud computing refers to the use of computer technology to deploy dynamically scalable and virtualized resources are provided as a service over a distributed network such as the internet. There are two several types of cloud environment of which the public cloud and the private cloud is the most popular. A public cloud is a network of services open to the general public for use free of cost. Some real time implementations include the Amazon EC2 [18], Google App Engine. A private cloud is an infrastructure owned by a single organization which can be managed either internally or by third party. A community cloud shares the infrastructure between several organizations working as a community with common goals. A hybrid cloud is a combination of two or more cloud environments such as the public, private and community clouds. This type of cloud offers the benefits of multiple deployment models. A personal cloud is an application of cloud computing implemented for an individual user.

Cloud offers a variety of services where SaaS, PaaS, IaaS are considered to be the most important of all and are the services that are widely used. IaaS is the service model which offers infrastructure components such as physical or virtual machines and other resources. Models such as Amazon EC2, Google Compute Engine, HP Cloud, Joynet, Navisite, RackSpace, and Windows Azure provide IaaS. PaaS is the service model that offers computing platform as a service. The platform includes operating system, the type of programming language used, environment the user executes, web servers, and database. AWS Elastic Beanstalk, Cloud Foundry, Heroku, Mendix, App Scale, OrangeScape, OpenShift offer PaaS. SaaS is the service model that offers software as a service. Users are provided with access to application software and databases. Google apps, Petrosoft, GT Nexus, Microsoft office 365 are some examples of SaaS.

There are no specific discovery mechanisms for searching different kinds of cloud services. Cloud consumers generally have to search for appropriate cloud services manually. Even though there are many existing generic search engines that consumers can use for finding cloud services, these engines may return URLs containing not relevant web-pages to meet the original service requirements of consumers [7]. Intuitively, visiting all the web-page can be time-consuming job. The following literature review discusses the various frameworks and technologies that can be used to develop an efficient and reliable search engine for searching cloud services.

In this paper we present a novel idea based on ontology which is an improvement over the standard way of finding the services manually. The concepts are identified for each relation and produce an optimal list of services by assigning end user tasks to the relevant data centers and also minimize span time thereby outperforming the existing methods in terms of search space and convergence. Agent technology can be effectively used for splitting up tasks in a distributed shared resource environment using multi agent approach. We can ensure that the agents specialize in identification of service provider for servicing the user requests.

Three agents are considered for this project. Requestor Agent, a software program to find appropriate providers with the appropriate computing resources for executing the search requests, through the Mapper agent. Search Agent provides query processing, for requests form Mapper Agent and maintains information in service providers for better mapping; Mapper Agent connects agents and facilitates the matching of service requests

from users to appropriate computing resource of providers. It may also be used to cope up with unexpected events.

Our approach is scalable by using multiple broker agents. The primary goal of this paper is to identify the best service provider. Using multiple agents provides efficient search mechanism in cloud environment.

The following may be done,

- The make span time in the cloud environment is minimized
- Handling service request and information of the processing unit is made available by designing agents.
- With shared and combined resources helps out in producing an optimal schedule minimizing the average waiting time

The remainder of the paper work is organized as follows: Section 2 discusses related work, Section 3 explains the existing methodology and describes the proposed approach and its parameters used. Section 4 describes the standard firefly algorithm .Section 5 describes the proposed jumper firefly algorithm. Section 6 presents the experimental results and Section 7 concludes the paper with future work and finally the references of this paper are provided.

2. RELATED WORK

A. Agent approach

In this approach [4] the agent team is taken into account which looks after the work of processing the queries and returning a list of all services discovered. The main drawback of this approach is that when the agent fails it cannot proceed. The agent can differentiate in performing work. Execution becomes difficult if the agent fails.

An Agent based framework in cloud computing ensures scalability when resources are either exhausted or are unavailable. As agents are mobile and autonomous in nature they can easily communicate and transfer information regarding the resource availabilities and other statuses. A Mobile agent is process that can transport its state from one ambience to another, with its data intact, and is able to perform appropriately in the new environment [6].

B. Multi agent approach

In the multi agent approach three types of agents (Requestor agent, Mapper agent and Search agent) are used. Mapper agent connects requestor agents to search agents using the connection algorithm. This algorithm consists of 4 stages, Selection, Evaluation, Filtering and Recommendation. The Recommendation stage uses two approaches (circular approach and multicast approach) for making recommendations to the requestor agents that failed to be matched with the mapper agents [8].

A Multi-Agent approach in resource allocation provides adaptation to infrastructure changes and also allocation of resources according to several customer and provider preferences. A Job agent initiates the negotiation and selects the best proposal for the customer's interests. The Resource agent makes the proposal with the appropriate service provider and completes the negotiation [17].

C. Ontology and information retrieval

Gathering of Web services based on Ontology supports the application that is accessible through the internet. Clients are able to quickly and easily select the needed services, information about their providers and even learn of the current attributes of resources behind the service. The user retrieving information through normal search engines take much time i.e., time consuming process. Whereas a similar application, which is developed under semantic environment will takes very less time for retrieving information [11].

It provides limited environment to get the conceptualization regarding the user need as information retrieval is based on the keywords provided. An ontology-based IR model is designed to exploit the domain knowledge to support semantic search. A novel semantic search model integrating keyword and semantic based search is used to achieve the benefits of both the approaches [21].

3. PROPOSED SYSTEM

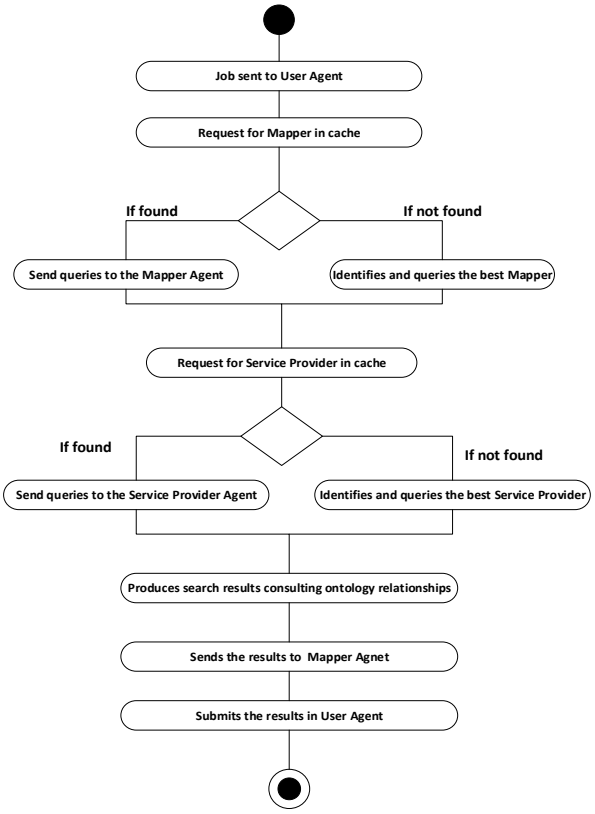
The idea of the proposed approach is to develop a multi agent based system that uses ontology relationships to find the similarities between the user queries and the available resources form various service providers. The system involves the use of agent approach as agents are autonomous and mobile so that they can easily communicate with each other over a distributed environment. The proposed approach illustrates the prototype of a cloud service discovery system that consists of three different agents - Requestor Agent, Mapper Agent, and Search Agent - and a search engine. The agents are developed using Java Agent Development Framework. The search engine uses a concept based search using similarity reasoning from the ontology relationships of the concepts in the cloud ontology.

To run a query is the biggest challenge in service requirement phase against the cloud services registered in the search engine's database if the consumer's budgetary and functional requirements are matched. When building a general search engine (e.g., Google), one only needs to consider the issue of searching for webpages that contain concepts in a user's query. Since one need to search for services that fulfill three requirement types, building search engine for cloud services becomes complex. The search engine in this work employs three different agents - Requestor agent, Mapper Agent, Search Agent - that consult cloud ontology for determining the similarities between providers' functional and technical specifications of services and consumers' functional and technical requirements.

The main objective of the proposed approach is to reduce the average waiting time for the searching of the

requested services. Also the overall waiting time of all the users is also tried to be reduced. This is achieved by the use of multiple agents in the architecture of the Cloudle search engine whereby each of the three agents specialize only in a specific function of capturing the request, mapping the request and retrieving the results for the request respectively.

It can be explained with the help of the activity diagram in Figure-1 as below,



4. CLOUDLE-THE SEARCH ENGINE

This section describes the Cloudle system which consists of three agents, Cloud ontology and a web interface. Cloudle is an agent-based search engine that consults Cloud ontology for reasoning about the relations of Cloud services. Inputs to Cloudle include service requirements from consumers (e.g., service name, OS type, CPU speed, etc.) and the outputs of Cloudle are lists of Cloud services ordered by the rating of their appropriateness to satisfy consumer's requirements. Rating of the appropriateness of Cloud services is determined by a Cloud service discovery agent that reason about Cloud services by consulting Cloud ontology.

The Cloudle system operates as follows. Cloud providers advertise their Cloud services by registering their services into the database of Cloudle. Queries are sent to the cloudle system using a web interface if the user is currently using any cloud system. The three agents carry out three main functionalities by consulting the Cloud ontology as follows.

Query processor: When a user sends a query to Cloudle, the query processor analyses the query and converts the query into an appropriate form which can be processed by the reasoning procedure.

Reasoning: In the reasoning procedure, the similarity between two Cloud concepts is determined by consulting Cloud ontology. The similarity between the two concepts is found using the similarity reasoning.

Rating: In the rating procedure, an aggregated similarity (service utility) is determined and ordered from highest to lowest service utility. A Cloud service which has the highest service utility would be selected and other services which have high service utility would be selected as well in the resulting web-page.

5. THE MULTI AGENT APPROACH

Some of the essential characteristics of cloud computing include resource pooling and resource sharing. In cloud, to serve multiple consumers, pooling of computing resources is made available. Also, a broad group of cross enterprise and cross platform users uses the available data and its applications.

Pooling and sharing of resources involve 1) co-operation between cloud providers by combining available resources. 2) Coordinating the resources shared among the cloud providers and scheduling and mapping the same. 3) Signing of a contract that is established between consumers (users) and providers.

In agent-based cloud computing, in order to automate the resource pooling and sharing possibilities in cloud, progress is done in cooperation and by co-ordinating protocols.

With the help of three different agents Cloudle is able to perform the search of a particular cloud service more effectively than a standard search engine. The Requestor Agent is primarily used to receive all the requests from multiple users simultaneously and stores it in its cache. The secondary function of the Requestor Agent is to forward the results of the user query once they are forwarded by the Search Agent which process the user query. The Mapper Agent acts as a mediator between the Requestor Agent and the Search Agent. The Mapper Agent takes the requests from the Requestor Agent and forward it to an available Search Agent to process the user query. Both the Requestor Agent and the Search Agent communicate with each other through the Mapper Agent. The Search Agent is the one that actually process the user query and returns the results. The Search Agent consults the cloud ontology to determine the similarities between the user queries and the services registered with the Cloudle's database. Once the results are generated the ordering of the results is based on the similarity value between the service required by the user and the available

service in the database. Finally the results are forwarded to the Requestor Agent to produce the results to the user.

6. SERVICE REASONING WITH ONTOLOGY

Given that Cloudle needs to satisfy three types of requirements - functional, technical, and budgetary - sometimes, it may be difficult to find services that will exactly match these three types of requirements. Similarity reasoning is designed so that finding relevant alternatives for any service is made easier. For instance, if some exact matching service found by the user is beyond a consumer's price ranging, other services that are similar to the user requirement within the limited price range may be suggested. An intuitive way to decide the degree of similarity between two concepts x and y is to determine how much x and y share in common. In similarity reasoning, the Agents determines the similarity between x and y by counting their common reachable nodes.

A. Similarity reasoning

Let α(x) (respectively, α(y)) be the set of nodes upwards reachable from x (respectively, y) including x (respectively, y) itself. For example, in the graph of general ontology (Figure-6.2), α(x) = 4 and α(y) = 3. Let α(x) ∩ α(y) be the number of reachable nodes shared by x and y. α(x) ∩ α(y) is a measure of the common features between x and y. In Figure-2, α(x) ∩ α(y) 2. There are several ways to define a function for determining the degree of similarity between x and y. One way to measure the degree of similarity between x and y is to measure the number of common features between x and y from the perspective of x as follows:

$$Sim(x,y) = \left| \frac{\alpha(x) \cap \alpha(y)}{\alpha(x)} \right|$$

Another way is to measure the number of common features between x and y from the perspective of y as follows:

$$Sim(x,y) = \left| \frac{\alpha(x) \cap \alpha(y)}{\alpha(y)} \right|$$

Figure-1 shows a simple ontology of operating systems [1]. In this example, the problem is to determine the similarities 1) between *Unix* and *Windows* and 2) between *Unix* and *Linux*. From Figure-6.1, it can be seen that α(*Unix*)=2, α(*Windows*)=2, α(*Linux*)=3, α(*Unix*)∩α(*Windows*)=1, and α(*Unix*)∩α(*Linux*)=2. For the purpose of experimentation, ρ is set to 0.5. Thus, *sim*(*Unix*, *Windows*)=0.5(1/2)+0.5(1/2)=0.5 and *sim*(*Unix*, *Linux*)=0.5(2/2)+0.5(2/3)=0.83. Hence, the *SDA* infers that compared to *Windows*, *Linux* is more similar to *Unix*.

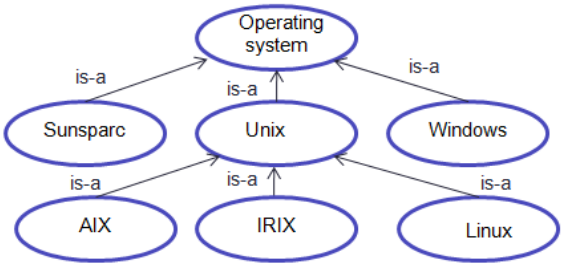


Figure-1. Simple ontology of operating systems.

B. Equivalent reasoning

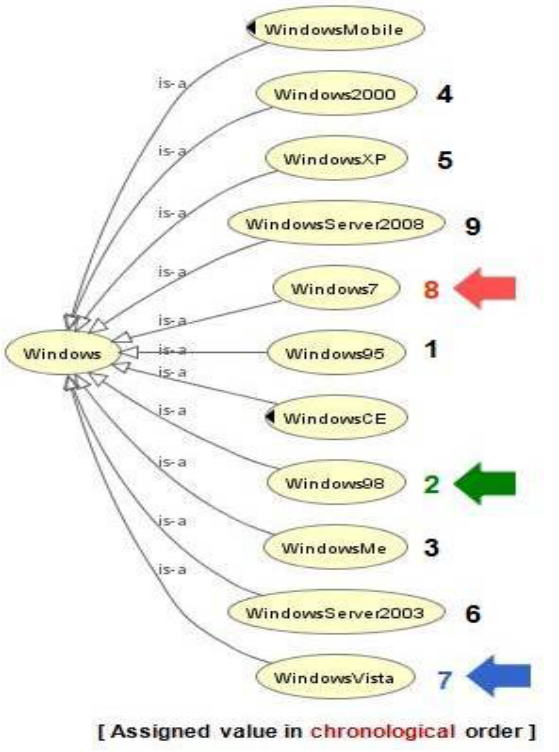


Figure-2. Simple cloud ontology of windows.

In Figure-2, the values 1 to 9 represent the chronological orderings of Windows. Windows95 is assigned 1 to indicate that it is the oldest version of Windows and WindowsServer2008 is assigned 9 representing the latest version. The compatibility between Windows98 and WindowsVista, and that between Windows7 and WindowsVista can be determined as follows.

It can be seen from Figure-2 that α(Windows98)=α(Windows7)= α(WindowsVista)=2, α(Windows98)∩α(WindowsVista=1 α(Windows7)∩α(WindowsVista)=1, sim(Windows98,WindowsVista)0.5, sim(Windows7,WindowsVista)=0.5

The label values of Windows98, WindowsVista, and Windows7 are cw98= 2, cwv= 7, and cw7=8. For the

purpose of experimentation, θ and μ will be 10 and 0.8 respectively.

$$Sim = Sim(x, y) + \frac{(0.8^{|Win98 - Win7|})}{10} = 0.53$$

$$Sim = Sim(x, y) + \frac{(0.8^{|Win7 - WinVista|})}{10} = 0.58$$

The search agent infers that compared to Windows98, Windows7 is more compatible with Windows Vista.

C. Numerical reasoning

Numerical reasoning is used for determining the similarity between two numeric concepts based on those label values. The similarity between two numeric values in same domain can be calculated as following formula.

$$Sim(a, b, c) = 1 - \left|\frac{a - b}{Max_c - Min_c}\right|$$

where a denotes the first numeric value and b denoted the second numeric value and c denotes the concept name.

Suppose that a consumer's CPUSpeed requirement is 3.0 and a service provider's CPUSpeed specification is 2.3. The similarity of their CPUSpeed specifications is determined as follows:

$$Sim(3.0, 2.3, cpu\ speed) = 1 - \left|\frac{3.0 - 2.3}{3.0 - 0.1}\right| = 0.86$$

7. EXPERIMENTAL STUDY

A series of experiments was carried out using the test system. Services are searched with the Cloud service name and the result from the searched services is selected. If the searched services are related to the Cloud service, the service utilities of these services are determined and the results are ordered based on the aggregated similarity. If the searched services are not related to Cloud service, the service utilities of these services are assigned zero, which means that the discovery failed.

Table-1. Experimental settings.

Experiment variables	**Value** (range)
# of providers	40
# of services for each provider	5 ~ 15 services
CPU clock	2.0 ~ 6.0 GHz
Ram size	0.256 ~ 36.0 GB
HDD size	3 ~ 8 TB
Network bandwidth	0.1 ~ 6 Gbps
Network latency	1~3000 ms

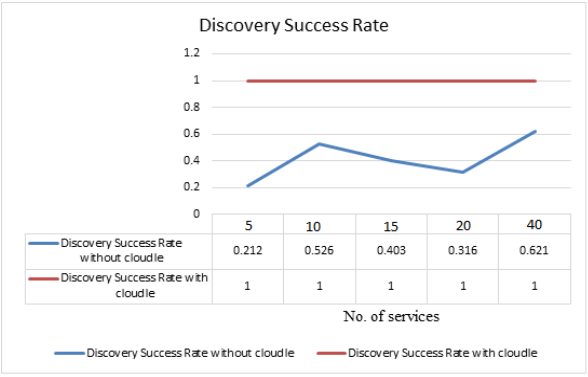


Figure-3. Discovery success rate.

Discovery success rate is determined by the ratio of the number of successes over the number of attempts. The result of using Cloudle with the Cloud ontology shows better performance than the results of using Cloudle without the Cloud ontology in terms of success rate. Furthermore, the result of using Cloudle with the Cloud ontology achieve 100% success rate. Based on the above observations, a user can find more appropriate Cloud services using Cloudle with the Cloud ontology.

CONCLUSION AND FUTURE WORK

The use of agents and ontology relationships in the mechanism of search engines has greatly changed the way the conventional working of the search engines. As multiple agents are used the average waiting time of a search is reduced also the overall waiting time an individual has also been reduced. The use of ontology has increased the chances of finding an exact match for the services requested. This is because all the service requirements are considered for searching the service and the service utility for each of the identified result is computed. With Cloudle, irrelevant web-pages are filtered out and with the Cloud ontology; the web-pages of Cloud service are rated by reasoning about the relations among Cloud concepts. Hence, the web-pages of the Cloud service have higher chance to be selected and are more likely to be closer to the requirements of user. At present, there are few Cloud service providers and there may not be many Cloud services available. However, when Cloud computing is more widely and commonly used in the near future, Cloudle can be helpful tool for Cloud users for finding Cloud services under their specific preference.
As an extension of this work the following may be done in future, adding more number of service providers and services into the cloud database to meet the changing user requirements, adding more concepts to the cloud ontology to differentiate two concepts more effectively.

REFERENCES

[1] Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu. 2008. Cloud Computing and Grid Computing 360-Degree

Compared‖ IEEE Grid Computing Environments (GCE 2008), Texas. pp. 1-10.

[2] Lamia Youseff, Maria Butrico, and Dilma Da Silva. 2008. Toward a Unified Ontology of Cloud Computing. Grid Computing Environments Workshop, GCE 2008.

[3] Phillip C-Y Sheu, Shu Wang, Qi Wang, Ke Hao and Ray Paul. 2009. Semantic Computing, Cloud Computing, and Semantic Search Engine. IEEE International Conference on Semantic Computing.

[4] Alexander Maedche, Steffen Staab. 2001. Ontology Learning for the Semantic Web. IEEE Intelligent Systems. 16(2): 72-79.

[5] Heiner Stuckenschmidt. 2002. Ontology-based information sharing in weekly structured environments. Ph.D. thesis, AI Department, Vrije University Amsterdam.

[6] FIPA 2001, Foundation for intelligent physical agents. FIPA Ontology Service Specification, http://www.fipa.org/specs/fipa00086/XC00086D.html .

[7] Dieter Fensel. 2003. Ontologies: A silver bullet for knowledge management and electronic commerce, Springer.

[8] D. Fensel. 2000. Understanding, Developing and Reusing Problem- Solving Methods, to appear as Lecture Notes of Artificial Intelligence (LNAI), Springer-Verlag, Berlin.

[9] M. R. Genesereth. 1991. Knowledge Interchange Format. In: Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR- 91). J. Allen *et al*., (eds). Morgan Kaufman Publishers. Pp. 238-249.

[10] T. R. Gruber. 1993. A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition. 5: 199-220.

[11] D. B. Lenat and R. V. Guha. 1990. Building large knowledge-based systems. Representation and inference in the Cyc project, Addison-Wesley, Reading, Massachusetts.

[12] Troels Andreasen, Henrik Bulskov, Rasmus Kanppe. 2003. From Ontology over Similarity to Query Evaluation. 2nd International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems (ODBASE), 3-7 November.

[13] Jun Wu, Xin Xu, Pengcheng Zhang, Chunming Liu. 2011. A novel multi-agent reinforcement learning approach for job scheduling in Cloud computing, Future Generation Computer Systems. Vol. 27, Issue 5, May. pp. 430- 439.

[14] Kang, Jaeyong Sim, Kwang. 2012. A multiagent brokering protocol for supporting Cloud resource discovery, Applied Intelligence. Springer Netherlends, 12 April. pp. 1-16. ISSN 0924669X, 10.1007/s10489-012-0347-y. (http://dx.doi.org./10.1007/s10489-012-0347-Y).

[15] L. Han and D. Berry. 2008. Semantic-supported and Agent-based Decentralized Grid Resource Discovery. Future Generation Computer Systems. vol. 24, no. 4, pp. 806–812, April.

[16] M. Miller. 2009. Cloud Computing: Web-Based Applications that Change the Way You Work and Collaborate Online. Que.

[17] I. Foster *et al*. 2008. Cloud Computing and Grid Computing 360- Degree Compared. Proc. Grid Computing Environments Workshop (GCE '08), pp. 1-10, November.

[18] R. Buyya *et al*. 2009. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation Computer Systems. vol. 25, no. 6, pp. 599-616, June.

[19] K.M. Sim. 2009. Agent-Based Cloud Commerce. Proc. IEEE Int'l Conf. Industrial Eng. and Eng. Management. pp. 717-721.

[20] M. Wooldridge. 2009. An Introduction to Multiagent Systems. second ed. John Wiley and Sons.

[21] K.M. Sim. 2010. Towards Complex Negotiation for Cloud Economy. Proc. Int'l Conf. Advances in Grid and Pervasive Computing (GPC '10). R.S. Chang *et al*., eds. pp. 395-406.

[22] T. Andreasen, H. Bulskov, and R. Kanppe. 2003. From Ontology over Similarity to Query Evaluation. Proc. Second Int'l Conf. Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems (ODBASE), November.

[23] L. Han and D. Berry. 2008. Semantic-Supported and Agent-Based Decentralized Grid Resource Discovery. Future Generation Computer Systems, vol. 24, no. 4, pp. 806-812, April.