*VFAST Transactions on Software Engineering*
*http://vfast.org/journals/index.php/VTSE@ 2013 ISSN(e): 2309-3978;ISSN(p): 2411-6246*
*Volume 1, Number 1, January-December, 2013*

*pp. 19-24*

# FORMAL MODELING OF AGENT BASED CLOUD COMPUTING SERVICES USING PETRI NETS

WASEEM IQBAL AND SHAHID YOUSAF
Department of Computer Science & IT
The University of Lahore,
Pakistan

Revised July 2013

ABSTRACT. *Cloud computing provides the better resource sharing and utilization of services through virtual shared servers. The allocation and deallocation of resources effects on quality, processing, memory, and service provisioning. The automation of cloud computing services is a big issue till now This model provides the better utilization of resources on demand through an agent by using formal method technique Petri net. Petri net modeling is a straightforward description of the problem which is used to overcome this type of issues. The agent acts as a dealer and responsible to utilize the unused resources, automation of resources, and provisioning of all types of cloud services like IaaS, PaaS, SaaS and HaaS. This Petri net modeling facilitates the utilization of resources through an agent by using predefined procedures. Furthermore, it also provides the efficiency and cost effectiveness of cloud computing services.*

***Keywords****:* Agent; Cloud computing; Petri net modeling; Resource utilization; Service provisioning.

1. **Introduction.** Cloud computing is an emerging technology and provides distributed computing through Internet services. Cloud computing provides dynamically scalable infrastructure or virtualized resources in the form of services over the Internet. It is a model for enabling scalable, on demand network access to a shared pool of configurable computing resources that can be provisioned ubiquitously and released with minimal management effort and cloud service provider interaction [1].

A dedicated runtime environment (e.g., supercomputer, cluster) has many advantages. Yet, its scalability is restricted by the available computing resources (i.e., number of processors in a supercomputer or number of nodes in the cluster). That is why nowadays we observe a migration of services, software or even a whole computing infrastructure into “clouds” (like Windows Azure6, Amazon EC27, or Google App Engine [2].

The challenge in cloud service negotiation is to establish SLAs between consumers and brokers, and between brokers and service providers. Whereas negotiation mechanisms involve two types of participants (buyers and sellers) in only one market and participants are not allowed to breach contracts, the problem of devising a complex negotiation mechanism for cloud computing is much more complex because a complex cloud negotiation mechanism specifies parallel negotiation activities among three types of participants (consumers, brokers, and providers) in multiple interrelated markets and participants are allowed to breach contracts by paying penalty fees. This work devises a complex cloud negotiation mechanism by using negotiation strategies and protocols in multiagent systems[3]. Cloud providers try to manage the huge traffic by configuring their sophisticated infrastructure to achieve their goals. They use their good strategy and planning for the management of applications, software’s and hardware’s. Cloud providers and other organizations that use datacenters are much concerned about configurations of their infrastructure. Commonly,

they use the faultless storage area network (SAN) technology for the configurations of large scale storage. Large scale storage is facing great challenges

In fact, the cloud computing users are increasing everyday and needs large scale infrastructure for high performance computing. Today Clouds are mainly used for handling highly intensive computing workloads and for providing very large data storage facilities. Both these goals are combined with the third goal of potentially reducing management and use costs. At the same time, multi-agent systems (MAS) represent another distributed computing paradigm based on multiple interacting agents that are capable of intelligent behavior. Multi-agent systems are often used to solve problems by using a decentralized approach where several agents contribute to the solution by co-operating with each other. One key feature of software agents is the intelligence that can be embodied in them according to some collective artificial intelligence approach that needs cooperation among several agents that can run on a parallel or distributed computer to achieve the high performance needed for solving large complex problems keeping execution time low. Although several differences exist between Cloud computing and multi-agent systems, they are two distributed computing models, therefore several common problems can be identified and several benefits can be obtained by the integrated use of Cloud computing systems and multi-agents. The research activities in the area of Cloud computing are mainly focused on the efficient use of the computing infrastructure, service delivery, data storage, scalable Virtualization techniques, and energy efficiency. In summary, we can say that in Cloud computing the main focus of research is on the efficient use of the infrastructure at reduced costs. On the contrary, research activities in the area of agents are more focused on the intelligent aspects of agents and on their use for developing complex applications. Here the main problems are related to issues such as complex system simulation, adaptive systems, software-intensive applications, distributed computational intelligence, and collective earning. Despite these differences, Cloud computing and multi-agent systems share several common issues and research topics in both areas have several overlaps that need to be investigated. In particular, Cloud computing can offer a very powerful, reliable, predictable and scalable computing infrastructure for the execution of multi-agent systems implementing complex agent-based applications such when modeling and simulation of complex systems must be provided. On the other side, software agents can be used as basic components for implementing intelligence in Cloud computing systems making them more adaptive, flexible, and autonomic in resource management, service provisioning and running large-scale applications [4].

A cloud service life cycle consists of: service requirements, service discovery, service negotiation, service composition, and service consumption [5]. The allocation strategy in private clouds, compared to a normal cloud, demonstrated a 87% reduction in energy consumption. It was observed that this strategy is not effective in scenarios where the workload is oscillating. That’s because it ends up generating too much unnecessary reconfigurations and migrations. Despite this, it still shows a significant gain in energy savings when compared to a cloud without any strategy deployed. The hybrid strategy for provisioning in green clouds, demonstrated a 52%  In the earlier period different agents support specific architectures, libraries, and protocols such as JACK, 3APL, Jason, Claim, JADE, SyMPA, ZEUS, Jadex, and Cougaar [6].

Different environments, toolkits, and methodologies of multi agent systems have been implemented to utilize the traditional computing systems and applications. It can be available for p2p, grid, and cloud networks for large scale multi agent systems to achieve the performance and scalability in distributed environment [7]. Till today only a few models, systems, techniques, and applications have been made for the integration of both cloud computing infrastructures and agent based technology [8]. In this paper we use Petri nets to model the system to overcome the problem like deadlock and unboundedness which remain present in the traditional specification. The Petri net is both graphical and analytical method the specify and verify the software and hardware systems used in many applications [].

**2.Cloud to Client Service Provisioning using Petri net.**

Petri net model of for cloud to client services is given in the figure 7.1. This graphical model illustrates that how agent based cloud provides the services to the clients. This net is composed of '*Places'* which are actually the condition/ services, '*Transitions'* which is the events '*Flow or Arc'* which tells the flow of the system and the last is '*Token'* which show the initial configuration of the Petri nets model. In the net given below p1 place contains the dot which is actually a token. To fire the transition t1 there must be a token in the place p1. p1 place represents the cloud in this net. Similarly p2, p3, p4, p5 places represent the services SaaS, PaaS, IaaS and HaaS of agent based cloud computing. Place p6 represent the client and t1, t2, t3, t4, t5, t6 represents the conditions, if these conditions are true means the transition are fired and token move to the

next place or places according to the situation.

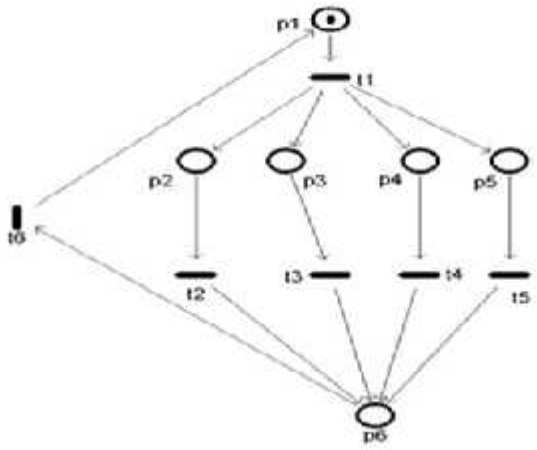


Figure 7.1: Cloud to client services.

For example initially the token is in the p1 place and only the transition t1 is fired and when we fired the this transition the token moves to the p2, p3, p4 and p5 places, which actually represents the cloud computing services means the services SasS, PaaS, IaaS and HaaS. Now the firing of t2 means that SaaS service is provided to the client. Similarly the whole networks. Following table shows the cover ability tree in text mode which shows how the transition are fired and which marking comes after firing which transition.

| Coverability Tree - Text Mode | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| From | Fired | To | From | Fired | To | From | Fired | To | From | Fired | To | From | Fired | To | From | Fired | To |
| M0 | t1 | M1 | M1 | t2 | M2 | M1 | t3 | M3 | M1 | t4 | M4 | M1 | t5 | M5 | M2 | t3 | M6 |
| M2 | t4 | M7 | M2 | t5 | M8 | M3 | t2 | M9 | M3 | t4 | M9 | M3 | t5 | M10 | M4 | t2 | M7 |
| M4 | t3 | M8 | M4 | t5 | M11 | M5 | t2 | M9 | M5 | t3 | M10 | M5 | t1 | M11 | M6 | t4 | M12 |
| M8 | t5 | M13 | M7 | t3 | M12 | M7 | t5 | M14 | M8 | t3 | M13 | M8 | t4 | M14 | M8 | t2 | M12 |
| M9 | t5 | M15 | M9 | t2 | M13 | M10 | t1 | M12 | M11 | t2 | M14 | M11 | t3 | M15 | M12 | t5 | M16 |
| M13 | t4 | M16 | M14 | t3 | M16 | M15 | t2 | M16 | M16 | t6 | M0 | | | | | | |
| M[p1,p2,p3,p4,p5,p6] | | | | | | | | | | | | | | | | | |
| M0 = [1,0,0,0,0,0] | | | | | | M1 = [0,1,1,1,1,0] | | | | | | M2 = [0,0,1,1,1,1] | | | | | |
| M3 = [0,1,0,1,1,1] | | | | | | M4 = [0,1,1,0,1,1] | | | | | | M5 = [0,1,1,1,0,1] | | | | | |
| M6 = [0,0,0,1,1,0] | | | | | | M7 = [0,0,1,0,1,0] | | | | | | M8 = [0,0,1,1,0,0] | | | | | |
| M9 = [0,1,0,0,1,0] | | | | | | M10 = [0,1,0,1,0,0] | | | | | | M11 = [0,1,1,0,0,0] | | | | | |
| M12 = [0,0,0,0,1,0] | | | | | | M13 = [0,0,0,1,0,0] | | | | | | M14 = [0,0,1,0,0,0] | | | | | |
| M15 = [0,1,0,0,0,0] | | | | | | M16 = [0,0,0,0,0,0] | | | | | | | | | | | |

Table 7.1: Coverability text mode tree for cloud to client service provisioning.

**2.1. Client with Cloud Requests for Service Provisioning.** In the second component of this architecture the client sends the request to the cloud for the services. The petri net model for this component is given in figure 7.2. In this Petri net model or net shows the flow that the client requests to cloud services to cloud by using multiple places p1, p2, p3, p4, p5, p6, p7, p8, p9, p10 and transitions t1, t2, t3, t4, t5, t6, t7. Here place p1 represents the client and p10 represents the cloud. Initially the token is in place p1 and t1 can fire after firing t1 token moves to the next places p2, p3, p4, p5 which means the client can send the request for any service like SaaS, PaaS, IaaS and HaaS. The rest of the configuration / simulation is illustrated in the given cover ability tree text mode.

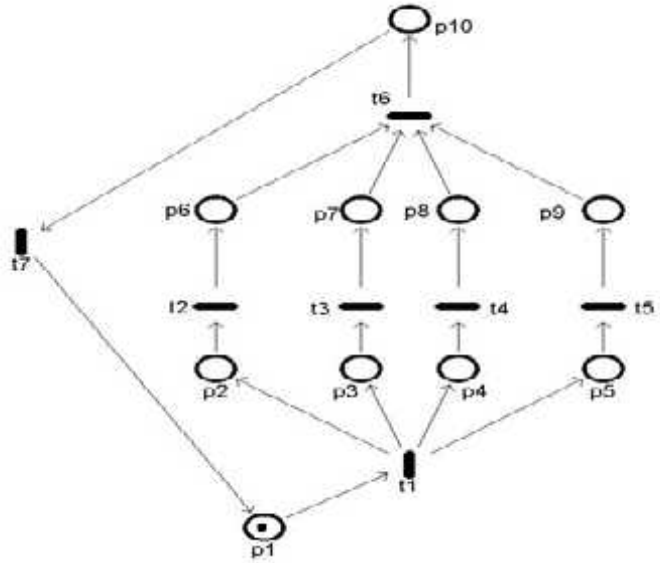


Figure 7.2: Client to cloud request.

Following is the coverability tree in text mode.

**Coverability Tree - Text Mode**

| From | Fired | To | From | Fired | To | From | Fired | To | From | Fired | To | From | Fired | To | From | Fired | To |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M0 | t1 | M1 | M1 | t2 | M2 | M2 | t5 | M3 | M1 | t4 | M4 | M1 | t5 | M5 | M2 | t3 | M6 |
| M2 | t4 | M7 | M2 | t5 | M8 | M3 | t2 | M9 | M3 | t4 | M9 | M3 | t5 | M10 | M4 | t2 | M7 |
| M4 | t5 | M9 | M4 | t5 | M11 | M5 | t2 | M9 | M5 | t3 | M10 | M5 | t4 | M11 | M6 | t4 | M12 |
| M6 | t5 | M13 | M7 | t3 | M12 | M7 | t5 | M14 | M8 | t3 | M13 | M8 | t4 | M14 | M9 | t2 | M12 |
| M3 | t5 | M15 | M10 | t2 | M13 | M10 | t4 | M15 | M11 | t2 | M14 | M11 | t3 | M15 | M12 | t5 | M16 |
| M13 | t4 | M16 | M14 | t3 | M16 | M15 | t2 | M16 | M16 | t6 | M17 | M17 | t7 | M0 | | | |

**M[p1,p2,p3,p4,p5,p6,p7,p8,p9,p10]**

| | | |
|---|---|---|
| M0 [1,0,0,0,0,0,0,0,0,0] | M1 [0,1,1,1,1,0,0,0,0,0] | M2 [0,0,1,*,1,1,0,0,0,0] |
| M3 = [0,1,0,1,*,0,1,0,0,0] | M4 = [0,1,1,0,1,0,0,1,0,0] | M5 = [0,1,1,*,0,0,0,1,0] |
| M6 = [0,0,0,1,*,1,1,0,0,0] | M7 = [0,0,1,0,1,1,0,1,0,0] | M8 = [0,0,1,*,0,1,0,0,1,0] |
| M9 [0,1,0,0,*,0,1,1,0,0] | M10 [0,1,0,1,0,1,0,1,0,0] | M11 [0,1,1,0,1,0,0,*,1,0] |
| M12 = [0,0,0,0,1,1,1,*,0,0] | M13 = [0,0,1,0,1,1,0,1,0,0] | M14 = [0,0,1,0,1,0,0,*,1,0] |
| M15 = [0,1,0,0,0,1,1,*,0,0] | M16 = [0,0,0,0,0,1,1,1,0,0] | M17 = [0,0,0,0,1,0,0,0,0,1] |

Table 7.2: Coverability text mode tree for client to cloud request provisioning.

**2.2. Cloud to Cloud Service Provisioning.** This component of the architecture is modeled with Petri nets is given in the figure 7.3. In the model the cloud can provide the services to other cloud. This net consists of four places p1, p2, p3, p4 and transitions t1, t2, t3, t4. Initially the token is in the p1 place. p1 place actually represents the cloud which provides the service to the second cloud which is p3. The following net and table illustrate it completely.

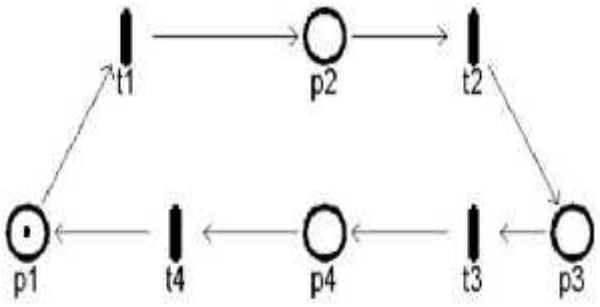


Figure 7.3: Cloud to cloud services.

Table given below shows the coverability tree in text mode for cloud to cloud service provisioning.

| Coverability Tree - Text Mode | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| From | Fired | To | From | Fired | To | From | Fired | To | From | Fired | To | From | Fired | To | From | Fired | To |
| M0 | t1 | M1 | M1 | t2 | M2 | M2 | t3 | M3 | M3 | t4 | M0 | | | | | | |
| M[p1,p2,p3,p4] | | | | | | | | | | | | | | | | | |
| M0 = [1,0,0,0] | | | | M1 = [0,1,0,0] | | | | | M2 = [0,0,1,0] | | | | | | | | |
| M3 = [0,0,0,1] | | | | | | | | | | | | | | | | | |

Table 7.3: Coverability text mode tree for cloud to cloud service provisioning.

**2.3. Cloud to Cloud Requests.** The last component of this architecture is cloud to cloud request where the cloud can send the request to the cloud for the services. Figure 7.4 shows the Petri nets model for this component of the architecture. This net consist of four places p1, p2, p3, p4 and four transition t1, t2, t3, t4. The place p1 represents the cloud which sends the request for services to the second cloud which is represented as p3. Here p3 is the cloud which sends the request to the p1 cloud for services.

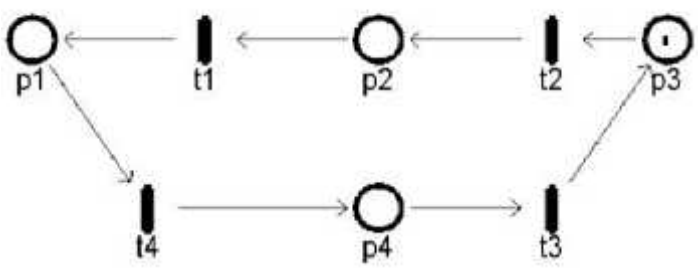


Figure 7.4: Cloud to Cloud Request.

Coverability tree in text mode for the cloud to cloud requests sending for services.

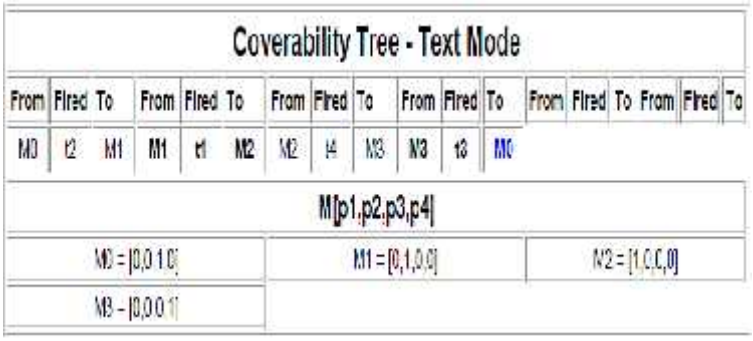


Table 7.4: Coverability text mode tree for cloud to cloud request provisioning.

**Conclusion**. In the paper we developed a formal model of agent based cloud computing by using Petri net. This model provides the better utilization of resources on demand through an agent by using formal method techniques. The agent acted as a dealer and responsibly to utilize the unused resources, automation of resources, and provisioning of all types of cloud services like IaaS, PaaS, SaaS and HaaS. The Petri net modeling is then used to verify the specification by coverability tree, shown deadlock freeness and boundedness.

## REFERENCES


[1]. Hada, P. S., Singh, R., & Manmohan, M. (2011). Security agents: a mobile agent based trust model for Cloud Computing. *International Journal of Computer Applications*, *36*(12).

[2]. Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008, November). Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08* (pp. 1-10). Ieee.

[3]. Sim, K. M., & Shi, B. (2010). Concurrent negotiation and coordination for grid resource coallocation. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *40*(3), 753-766.

[4]. Talia, D. (2011). Cloud Computing and Software Agents: Towards Cloud Intelligent Services. In *WOA* (pp. 2-6).

[5]. Joshi, K., Finin, T., & Yesha, Y. (2009, October). Integrated lifecycle of IT services in a cloud environment. In *Proceedings of The Third International Conference on the Virtual Computing Initiative (ICVCI 2009), Research Triangle Park, NC*.

[6]. Aversa, R., Di Martino, B., Rak, M., & Venticinque, S. (2010, February). Cloud agency: A mobile agent based cloud system. In *Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on* (pp. 132-137). IEEE.

[7]. Cao, B. Q., Li, B., & Xia, Q. M. (2009). A service-oriented QoS-assured and multi-agent cloud computing architecture. In *Cloud Computing* (pp. 644-649). Springer Berlin Heidelberg.

[8]. Lopez-Rodriguez, I., & Hernandez-Tejera, M. (2011). Software agents as cloud computing services. In *Advances on Practical Applications of Agents and Multiagent Systems* (pp. 271-276). Springer Berlin Heidelberg.

[9]. Khan, S. A., Ahmad, F., Fakhir, I., ur Rehman, M., & Ullah, M. (2012). Structural Analysis Methods for Petri Net based Control Systems: a Review.Journal of American Science, 8(12).

[10]. Ahmad, F., & Khan, S. A. (2012). Module-based architecture for a periodic job-shop scheduling problem. Computers & Mathematics with Applications, 64(1), 1-10.

[11]. Ahmad, F., & Khan, S. A. (2012). Specification and verification of safety properties along a crossing region in a railway network control. Applied Mathematical Modeling. 37(7),5162-5170