# Agent Based Application Tools for Cloud Provisioning and Management

Luca Tasquier, Salvatore Venticinque, Rocco Aversa, and Beniamino Di Martino

Department of Information Engineering, Second University of Naples, Aversa, Italy
luca.tasquier@gmail.com, {salvatore.venticinque,rocco.aversa}@unina2.it, beniamino.dimartino@unina.it

**Abstract.** When service providers move to IAAS Clouds, whether their service is delivered by a legacy application, either it has been developed by a deployable open platform, the provisioning and the management workflow of the computing infrastructure really changes. Here we describe a set of tools which can be used to orchestrate agents' based services, which provide facilities for provisioning, management and monitoring of Clouds. The user is able to discover, allocate, configure and monitor Cloud services at infrastructure level through an approach that is agnostic respect to the specific Cloud vendor or to the Cloud technology.

## 1 Introduction

When service providers move to IAAS Clouds, whether their service is delivered by a legacy application, either it has been developed by a deployable open platform, the provisioning and the management workflow of the computing infrastructure really changes. First of all the Cloud elasticity supports the reconfiguration of the computing resources when application requirements change dynamically and the *pay-per-use* business model allows for the possibility to change the Resource Providers when a more convenient offer is found. Of course a number of issues arise in the current Cloud scenario due to the lack of interoperability among different technological Cloud solutions and because of the limited portability of Cloud applications. However, even when the service developer is able to overcome these difficulties, by making technical choices that are independent respect to the Cloud provider, it is not easy to discover and retrieve the available Cloud proposals, to check if they can accomplish the service requirements, and also to compare each other. Currently there is not a common ontology for describing service terms and service levels, neither in a formal way nor through natural language. Other issues regard the management of the acquired resources. Also in this case the lack of a wide adopted standard for service at Cloud infrastructure level (IAAS) affects the chance of opting for a different commercial or technological solution. In fact the use will have to change both management tool and methodology. Finally, the last motivation we are addressing deals with monitoring of resource utilization. This problem

M. Yousif, L. Schubert (Eds.): Cloudcomp 2012, LNICST 112, pp. 32–42, 2013.

has been extensively investigated with the perspective of the resource provider, which aims at optimizing the utilization of its physical resources in order to improve its own service level and to increase its profit. However monitoring needs to be addressed with a different perspective in the case of a service provider that stocks computing resources through the Cloud market. Cloud customers cannot check the compliance of the Service Level Agreement (SLA) trusting the monitoring service of the same provider, who has a conflicting interest ensuring the guarantees on service levels it provides. Besides Cloud customers needs to detect under-utilization and overload conditions. In both the cases it is necessary to dimension the Cloud resource to avoid useless expenses and to not fail to satisfy the service requirements when workloads change dynamically. In this paper we present a set of tools which allows the user for orchestrating agents based services, which support discovery, brokering, management and monitoring of Cloud resources. We describe how these services can be used to execute a workflow for Cloud governance that allows for vendor agnostic provisioning, deployment, management and monitoring Cloud services at Infrastructure level. Related work is presented in Section 2. In Section 3 we introduce the available services and a workflow for Cloud governance. Section 5 describes application tools and how they can be used. Finally conclusions are due.

## 2 Related Work

The design and development of solutions for governance of multiple heterogeneous cloud is an issue addressed both in research activities and commercial domains [4]. Here we briefly provide an overview about related work and the technological assessment for Cloud provisioning, management and monitoring. The brokering of Cloud providers, whose offers can meet the requirements of a particular application, is a complex issue due to the different business models that are associated with such computing systems. The current Cloud computing technologies offer a limited support for dynamic negotiation of SLAs among participants. The work presented in [10] represents a first proposal to combine SLA-based resource negotiations with virtualized resources in terms of on-demand service provision. The architecture description focuses on three topics: agreement negotiation, service brokering and deployment using virtualization. It involves multi- ple brokers. A Cloud multi-agent management architecture is proposed in [5]. A simpler agents based architecture has been proposed in [13]. Preliminary investigations by the authors on related topics have been presented in [12]. SLA@SOI is the main project that aims (together with other relevant goals) at offering an open source based SLA management framework. It will provide benefits of predictability, transparency and automation in an arbitrary service-oriented infrastructure. About management of heterogeneous Clouds, interoperability is the main issue. By the research community there are many standardization efforts. Some examples are OCCI (Open Cloud Computing Interface), by the Open Grid Forum, and SOCCI (Service-Oriented Cloud-Computing Infrastructure) by the ISO Study Group on Cloud Computing (SGCC). In particular OCCI (Open Cloud Computing Interface) is a proposal of standard for

IAAS Cloud. It defines entities, relationships API and protocols for all kinds of management tasks. This solutions is aimed at the fulfillment of three requirements: integration, portability and interoperability for common tasks including deployment, autonomic scaling and monitoring still offering an high degree of extensibility. The OpenNebula solution already implements a RESTFull OCCI compliant interface, and other technologies like Eucalyptus and Openstack are working to be themselves compliant with it. Commercial providers are going to support themselves management facilities which enable the integration of third party clouds. For example Amazon can use its elasticity capability also exploiting computing resources shared by OpenNebula. Rather than adopting a standard or developing a new service interface, some efforts have been spent to develop technologies for integration of existing IAAS Clouds. DeltaCloud and JClouds are two different solutions. The first one provides a service with an uniform interface, but uses different drivers to redirect requests to the supported heterogeneous commercial Cloud providers and to private Clouds developed by open technologies. JCLouds, instead, offers a uniform and extensible API to develop applications that can directly interoperate with multiple IAAS Clouds. Another example of free and open source technology that aims at supporting the integrated management of heterogeneous Cloud is provided by My Cloud Portal[1]. It allows for setting up and managing of hybrid cloud, private and public, by the integration of Eucalyptus and Azure Cloud infrastructures. It provides a web interface by which it is possible to define workflows, perform monitoring activities and reconfigure settings. Infrastructure- level resource monitoring [6] [3] aims at the measurement and reporting of system parameters related to real or virtual infrastructure services offered to the user (e.g. CPU, RAM, or data storage parameters). Traditional monitoring technologies for single machines or Clusters are restricted to locality and homogeneity of monitored objects and, therefore, cannot be applied in the Cloud in an appropriate manner [8]. At the state of the art there are many tools which provide Cloud monitoring facilities, like Cloudkick, Nimsoft Monitor[2], Monitis[3], Opnet, RevealCloud. All of them are proprietary solutions and do not aim at defining a standard for monitoring interoperability. Some technologies for monitoring network and host like sFlow have been extended in order to support the transport of monitoring information of virtualized resources. For example host-sflow [1] exports physical and virtual server performance metrics by using the sFlow [2] protocol. Ganglia and other collectors of performance measures are already compliant with its specification. In [9] authors claims that an approach based on software agents is a natural way to tackle the monitoring tasks in the aforementioned distributed environments. Agents move and distribute themselves to perform their monitoring tasks. In [11] an optimal control of mobile monitoring agents in artificial-immune-system-based (AIS-based) monitoring networks has been studied.

[1] `http://www.mycloudportal.in/`
[2] `http://www.nimsoft.com/solutions/nimsoft-monitor`
[3] `http://portal.monitis.com`

## 3 Deployment and Execution Using the IAAS Cloud

As shown in Figure 1 the life cycle of a Cloud application that is running by using a Cloud infrastructure is divided into three phases:

1. *Cloud Provisioning.* The user has to choose the best Cloud resources for his/her application (e.g. VMs, storages, etc.). After that he/she has to select the best IaaS provider basing his/her thinking on a lot of parameters (cost per use, amount of VM memory, storage's size, bandwidth, etc.). Many times this reasoning is too difficult because each provider offers its resources highlighting different characteristics and parameters. This happens because the Cloud vendors haven not a common and standardized interface to describe the resource parameters, making the comparison among same resources an hard job.
2. *Cloud Configuration.* After selecting the best resources for his/her application, the Cloud customer needs to sign an SLA with a Cloud vendor. Once this has been done some management activities are carried before deploying applications. For instance OS images have to be attached and the purchased VMs have to be started. For this reason, the cloud user/developer has to know the allowed actions for that resource and the service interface for that Cloud provider. In fact the same resource purchased from a different provider have different interface and different supported functionalities. At this point the Cloud application can be deployed and executed.
3. *Cloud Monitoring.* Here Cloud users configure a network of probes that collects measures about the performance parameters of the Cloud resources. To get an up to date knowledge of Cloud performance and an history of the Cloud behavior it needs to periodically compute a set of performance indexes and to set up some triggers which notify critical conditions. In fact it would be useful to know if the workload of the infrastructure is different from the one foreseen, in order to avoid saturation or under-utilization of Cloud resources. This information is necessary to design effective reconfigurations of the infrastructure, in order to better adapt it to the current application requirements and to optimize performances and costs.

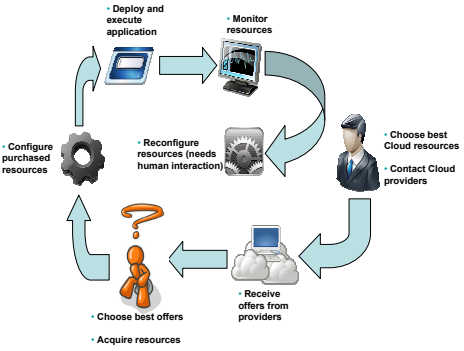



**Fig. 1.** Deployment and execution workflow by using IAAS Cloud

## 4 The mOSAIC Solution: Cloud Agency

The EC-FP7-ICT mOSAIC project [7] intends to improve the state-of-the-art in Cloud computing by creating, promoting and exploiting an open-source Cloud application programming interface and a platform targeted for developing multi-Cloud oriented applications. Cloud Agency represents the mOSAIC solution for provisioning and management of Cloud resources that allows for the deployment and execution of the mOSAIC platform and applications.

Cloud Agency [12] is a multi agent system that accesses, on behalf of the user, the utility market of Cloud computing to maintain always the best configuration of resources, to satisfy the application requirements. It supplements the common management functionalities, which are currently provided by Cloud infrastructures, with new advanced services, by implementing transparent layer to IAAS Private and Public Cloud services.

Agents can be orchestrated by invoking those services, which are offered by Cloud Agency through an OCCI compliant RESTFull interface, according to the workflow defined in Section 3.

One of the leading agents that composes Cloud Agency is the *Broker Agent* that receives the list of those resources that the mOSAIC application needs for its deployment and execution, asks to providers for available offers, brokers the best one and allows for closing the transaction. *Vendor Agents* implement a wrapper for a specific Cloud: they are used to get an offer for resource provisioning, to accept or refuse that offer, to get informations about a resource or to perform an action on it (start, stop, resume). About Cloud monitoring, *Meter Agents* perform measures of performance indexes at IAAS level and return their values to the *Archiver Agent*. The Archiver collects the measures and maintains statistics about the mOSAIC system. For the reconfiguration service the *Tier Agent* has been developed: it is triggered by the Archiver and uses policies defined by the user to apply the necessary reactions. The *Mediator Agent* receives requests directly from the user: it is in charge of starting new transactions for provisioning by creating Broker agents; it also starts new Tiers and Meters, and returns information about Cloud Agency configuration and services.

Being implemented as a MAS (Multi Agent System), the internal work-flow of the agency is intrinsically asynchronous because agents react on the occurrence of an incoming message. In the same way RESTFull APIs of Cloud Agency have been conceived for an event-driven mOSAIC programming model. Asynchronous Service Requests (ASR) are used to ask Cloud Agency for something to be executed. For example to start a negotiation, to accept or to refuse the SLA, to start a VM, etc. Requests, as any other action, are not-blocking. It means that execution is started remotely, but the client can continue to run. On the other hand, requests will generate future events, which have to be handled by the requester. By the same way, asynchronous events from Cloud Agency are notified to the Cloud user. Synchronous Service Requests (SSR) are used to get informations. For example clients can ask for reading an SLA or the status of a negotiation, to get the list of vendors or the list of the resources. Queries are synchronous, they return immediately the response if it is available, and an exception otherwise.

# 5 Application Tools

A set of tools have been designed and developed to support the orchestration of Cloud Agency services by different type of Cloud users. The orchestrator can be actually personified by both a human user and by an autonomic application. In the first case the user takes the hat of a Cloud administrator, in the second case it is a developer that design the application. The list of tools includes some APIs for development of legacy applications and mOSAIC Cloud applications. For sake of space we skip the description of mOSAIC APIs development here. Instead we present two implementations of a Cloud Agency console and a Monitoring tool.

## 5.1 Cloud Agency Client API

*A set of JAVA APIs* support the development of event based client application of Cloud Agency. According to the event-driven interaction model the API allows for sending asynchronous and synchronous requests and for handling asynchronous notifications by implementing callbacks. The class diagram of a typical client application is shown in Fig. 2.

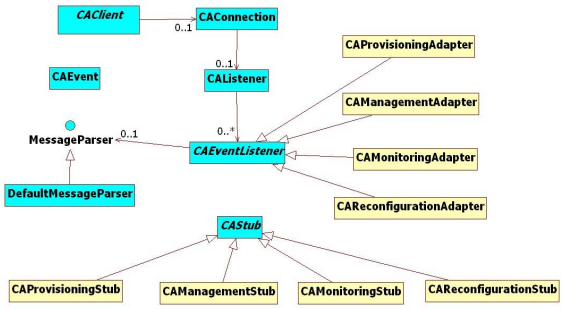



**Fig. 2.** Class diagram of Cloud Agency Client API

The blue colored classes represent the *core* API. The main goals of each class are described in Table 1.

In addition to the core API, several implementations of core abstract classes are provided. In particular:

– *stubs*: a stub offers a set of methods to invoke the Cloud Agency services. These methods wrap the REST requests in order to query the Cloud Agency RESTFull interface and to perform actions on Cloud resources or to obtain information about the state of the Cloud infrastructure. At the state-of-the-art of the development API client there are four classes that implement the CAStub class and that provide methods to invoke provisioning, management, monitoring and reconfiguration services;

**Table 1.** Core classes of Cloud Agency Client API

| Class | Description |
|---|---|
| CAClient | This abstract class represents an instance of the Cloud Agency Client. Who wants to develop a new client for the Cloud Agency has to extend it |
| CAConnection | This class represents the established connection between the client and the Cloud Agency. It is in charge of authenticating the user against the Cloud Agency and of initializing instances of stubs and adapters |
| CAEventListener | This abstract class implements a generic event listener. It handles the generic asynchronous messages coming from the Cloud Agency |
| CAEvent | This class abstracts an asynchronous event received from a listener |
| MessageParser | This interface represents the message parser used by an event listener in order to process an asynchronous message coming from the Cloud Agency and to generate a CAEvent |
| DefaultMessageParser | This is the default implementation of a MessageParser |
| CAStub | This abstract class implements the generic stub |

– *adapters*: an adapter is an handler for the asynchronous messages coming from the Cloud Agency, according to its event-driven architecture. When a new message arrives, it is forwarded to the adapter that implements the particular service listener. Each user can customize the client behavior on the arrival of particular events by simply extending the *CAEventListener* class or overriding an existing adapter. So he/she can implement autonomic reactions by adding his/her own new adapter by the core API. At the state-of-the-art of the development of the API client there are four classes that implement the *CAEventListener* class and that handle (without performing any action outside of the displaying the received message) provisioning, management, monitoring and reconfiguration events.

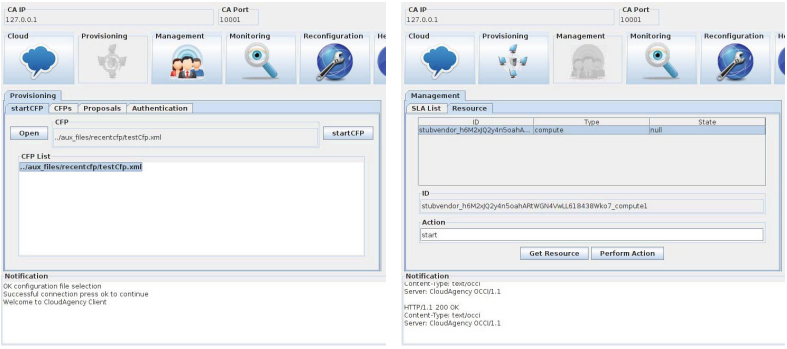
These APIs are also used to implement a command line and a graphical user interface of a Cloud Agency console.

### 5.2 Command Line and Graphical User Interface

Cloud Agency **Command Line Interface (CA-CLI)** offers a variety of commands to invoke the Cloud Agency's services. It follows the user in all the phases of the deployment and of the execution of his/her own application by giving him/her the possibility to book and manage the Cloud resources in a flexible and simple way. The CA-CLI helps the mOSAIC developer in the deployment process, beginning from the brokering of the best resources for his/her application. The application opens a console that starts a listener to handle the notifications sent by Cloud Agency and allows for the execution of a number

of commands. The management is vendor agnostic, in the sense that the user asks for performing a specific operation (start, stop, restart, etc.) on any given resource.

Cloud Agency **Graphical User Interface (CA-GUI)** is another tool that helps the user during the deployment and execution of his/her mOSAIC application. The functionalities offered by the CA-GUI are basically the same of the CA-CLI ones. Of course, the graphical interface is more powerful than the command line to take under control all the stuff during the provisioning phase: the editing and the listing of the *Call For Proposals (CFPs)*, the providers' proposals or SLAs, the acquired resources and their state. It also provides some additional functionalities in order to simplify the Cloud management, such as the listing of the available vendors, the start/stop of an available VM, the attach and detach of an available storage and so on.




(a) GUI tab for provisioning (b) GUI tab for management

**Fig. 3.** Graphical User Interface for Cloud Agency Client

In Fig. 3 (a) it is shown how the CA-GUI appears. On top of the interface there is the location of the connected Cloud Agency instance. Just below this one there are some buttons, each one representing a Cloud Agency service and allowing the user for easily performing provisioning, management, monitoring and reconfiguration operations. Moreover, the *Cloud* button provides several functionalities to manage the CFPs and to get informations about Cloud Agency status. By clicking a button, a new panel appears, which is customized by the particular operations that the selected service allows. The CA-GUI handle both ASR and SSR in order to get the requested informations and/or to perform an action. On the bottom of the window the raw notification messages are displayed.

The management console, shown in Fig. 3 (b) allows the user for starting/stopping VMs, for loading and attaching VM images, for deploying and executing applications and so on.

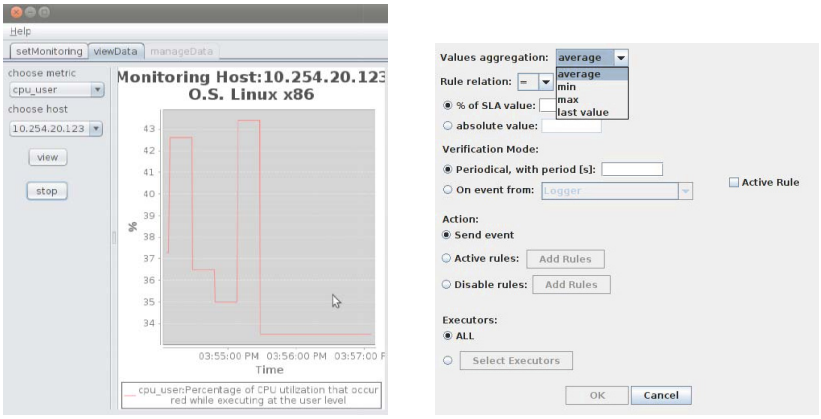
The monitoring console allows for the configuration of the monitoring infrastructure on the acquired resources. For each resource is possible to select a set of

measures, each one supported by a specific by Meter Agent. When a set of measures has been selected and the monitoring configuration is finished, the Cloud Agency creates a new Meter Agent sends it to the target resource. At this point the Meter Agent get the measures and sends them to the Archiver Agent, that stores them and is able to compute metrics on performace information on user's demand. Currently is possible to choose Host sFlow based measures. *Host sFlow* [1] agent measures and communicates physical and virtual server performance parameters using the *sFlow* [2] protocol. The agent provides scalable, multi-vendor, multi-OS performance monitoring with minimal impact on the systems being monitored.

### 5.3 Cloud Performance Monitor

After that the application has been deployed. As regards the mOSAIC developer, he/she can take under control the infrastructure performances by using the monitoring tool that can be started by the CA-GUI. It allows for the visualization of a list of available measures, as it is shown in Fig. 4 (a), or for setting up the computation of some metrics about performance indexes. When a new metric is created, the developer can read synchronously the last value of that index or can create a trigger to be notified asynchronously according to a specific time period or when a critical condition is verified as it is shown in Fig. 4 (b). An example of available metric is the average value of a measure that is periodically computed and that is notified when it is out of a certain range.

When a trigger is activated by the verification of a critical condition on a resource's parameter, the user can decide to be notified about the verified event or to activate/deactivate other rules previously defined and related to other




(a) Visualization of performance's indexes

(b) Creating a trigger on a resource's parameter

**Fig. 4.** Monitoring Tool for Cloud Agency

resource's parameters and/or other resources. So the developer can set up a complex trigger by composing some simple ones.

## 6 Conclusion

A set of tools for provisioning and management of Cloud resources can leverage the burden of a user that wants to deploy and execute his/her application by using services at infrastructure level. In the framework of the FP7 mOSAIC project we have designed and developed agents based services for provisioning, management and monitoring of Cloud infrastructure. We presented here an API client and some application tools that allow for the orchestration of such agents based services according to a workflow to be adopted for the governance of Cloud resources. The human intervention is still required for taking necessary decisions about if and how reconfigure the infrastructure by using the Cloud elasticity or looking for new proposals, eventually from different providers. Future work will focus on the design and developing of autonomic applications to be automatically triggered when critical conditions have been detected in order to provide an autonomous and configurable closed-loop implementation of the proposed workflow.

**Acknowledgements.** This research is partially supported by FP7-ICT-2009-5-256910 (mOSAIC) and by Cloud@Home PRIN project by MIUR.

## References

1. Host sflow, `http://host-sflow.sourceforge.net/`
2. sflow, `http://www.sflow.org`
3. Aversa, R., Di Martino, B., Venticinque, S.: Distributed agents network for ubiquitous monitoring and services exploitation, pp. 197–204 (2009) Cited By (since 1996)
4. Aversa, R., Di Martino, B., Venticinque, S.: Integration of mobile agents technology and globus for assisted design and automated development of grid services, pp. 118–125 (2009); Cited By (since 1996)
5. Cao, B.Q., Li, B., Xia, Q.M.: A service-oriented qos-assured and multi-agent cloud computing architecture. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) Cloud Computing. LNCS, vol. 5931, pp. 644–649. Springer, Heidelberg (2009)
6. Clayman, S., Galis, A., Chapman, C., Toffetti, G.: Monitoring service clouds in the future internet. Framework, 115–126 (2010), `http://www.future-internet.eu/fileadmin/documents/valencia_documents/plenary/Monitoring_service_clouds_in_the_Future_Internet.pdf`
7. Di Martino, B., Petcu, D., Cossu, R., Goncalves, P., Máhr, T., Loichate, M.: Building a mosaic of clouds. In: Guarracino, M.R., Vivien, F., Träff, J.L., Cannatoro, M., Danelutto, M., Hast, A., Perla, F., Knüpfer, A., Di Martino, B., Alexander, M. (eds.) Euro-Par-Workshop 2010. LNCS, vol. 6586, pp. 571–578. Springer, Heidelberg (2011)

8. Emeakaroha, V.C., Brandic, I., Maurer, M., Dustdar, S.: Low level metrics to high level slas - lom2his framework: Bridging the gap between monitored metrics and sla parameters in cloud environments. In: Smari, W.W., McIntire, J.P. (eds.) HPCS, pp. 48–54. IEEE (2010), http://dblp.uni-trier.de/db/conf/ieeehpcs/ieeehpcs2010.html#EmeakarohaBMD10
9. Ilarri, S., Mena, E., Illarramendi, A.: Using cooperative mobile agents to monitor distributed and dynamic environments. Inf. Sci. 178(9), 2105–2127 (2008)
10. Kertesz, A., Kecskemeti, G., Brandic, I.: An sla-based resource virtualization approach for on-demand service provision. In: Proceedings of the 3rd International Workshop on Virtualization Technologies in Distributed Computing, VTDC 2009, pp. 27–34. ACM, New York (2009)
11. Liu, W., Chen, B.: Optimal control of mobile monitoring agents in immune-inspired wireless monitoring networks. Journal of Network and Computer Applications 34(6), 1818–1826 (2011), doi:10.1016/j.jnca.2010.12.004
12. Venticinque, S.: Agent Based Services for Negotiation, Monitoring and Reconfiguration of Cloud Resources, pp. 178–202 (2012)
13. You, X., Wan, J., Xu, X., Jiang, C., Zhang, W., Zhang, J.: Aras-m: Automatic resource allocation strategy based on market mechanism in cloud computing. Journal of Computers 6(7) (2011), http://ojs.academypublisher.com/index.php/jcp/article/view/jcp060712871296