A Cloud Federation architecture

Based on Open Technologies

Gianluca Zangara, Diego Terrana, Pietro Paolo Corso
Department of Physics and Chemistry
University of Palermo
Palermo, Italy
gianluca.zangara@unipa.it, diego.terrana@unipa.it,
pietropaolo.corso@unipa.it

Marco Ughetti, Guido Montalbano
TILAB
Telecom Italia S.p.A.
Torino, Italy
marco.ughetti@telecomitalia.it,
guido.montalbano@telecomitalia.it

***Abstract*— Cloud Computing is the state of the art computation paradigm enabling providers to offer computing resources to customers in a pay-per-use fashion. Nowadays public Cloud providers offer similar services within few service models, mainly IaaS and PaaS. Cloud providers give to the user the feeling to dispose of infinite resources, thus having to predict the user requirements in order to provide services with minimal costs, maintaining at the same time high levels of SLAs. In order to achieve this goal, Cloud providers can cooperate together to bring new business opportunities, such as expanding available resources, achieving cost effective asset optimization and adopting power saving policies. Cloud Federation allows different Cloud providers the opportunity to work collaboratively to offer best services to customers and contemporary to improve their productivity. Customers can advantage from Cloud Federation for a larger offer of available services, the capability of price comparison and the removal of vendor lock-in. In this paper we describe a platform that enables the federation of several heterogeneous Cloud Providers to allow the customers choosing and activating Cloud services from a central platform, bringing more attractive price policy to customers. The authors introduce a prototype of Cloud Federation platform based on a central infrastructure tested to manage OpenStack, CloudStack and Amazon EC2 providers, thus allowing the user to select the best services in terms of either technical requirements or price policy and activate them without having to explicitly register to each of the federated providers. The prototype is designed to accept different types of Cloud service models by means of transparent interfaces developed around a billing and a metering module, respectively to bill the service to the customer and to collect information about the health status of the federated platforms.**

Keywords — Cloud computing; Cloud federation; OpenStack; CloudStack

I. INTRODUCTION

In the last decades a real revolution has affected the ICT world driven by many different factors: the diffusion of broadband connections and personal devices; the advent of resource virtualization and the possibility of using generic resources in an on-demand manner, not necessarily on-premise; the ever-increasing production of a huge amount of structured and unstructured data containing valuable information to be analysed.

All these factors have determined a real revolution in the habits of the users and in the market offer of new ICT products specifically designed on them. At the same time, traditional contexts like High Performance Computing (HPC), till 90s considered accessible only within dedicated and isolated computing centres, usually devoted to scientific research, have opened to federation capabilities by means of the sharing of the computational resources. This scenario marked the advent of Grid Computing (GC) based on the possibility of sharing homogeneous pools of resources. The advent of resource virtualization, started by CPU virtualization and then extended to all kind of devices and resources (storage, networks, applications), has opened a whole new world allowing the access not only to specific resources, used at a low level, but also to entire and complete platforms and systems, offered on-demand, according to specific and detailed user needs. Virtualization allows the partitioning of computer resources into multiple virtual units thus allowing to use these partitions as they were physical computers.

Cloud Computing (CC) represents the natural evolution of the above mentioned technologies offered in an almost completely transparent way for the end user who can access to a wide range of resources (CPUs, storage, network, applications, etc.) offered in an elastic way by a service provider. The organizations that make use of Cloud technologies can therefore access to computing resources offered by external providers only when they need and using the exact amount of resources they need to satisfy a specific need.

The great potential of CC manifests itself through the offer of a wide range of services as summarized by the following three approaches (Figure 1):

- Infrastructure-as-a-Service (IaaS) – basic computing resources (i.e. CPUs, RAM memory, storage, networking) are offered by a provider as virtual instances;
- Platform-as-a-Service (PaaS) – a complete computing platform is offered as an application middleware;
- Software-as-a-Service (SaaS) – the cloud provider installs and operates application softwares and

978-1-4673-9473-4/15 $31.00 © 2015 IEEE
DOI 10.1109/3PGCIC.2015.183

databases accessible by and customised over the needs of the end-user.

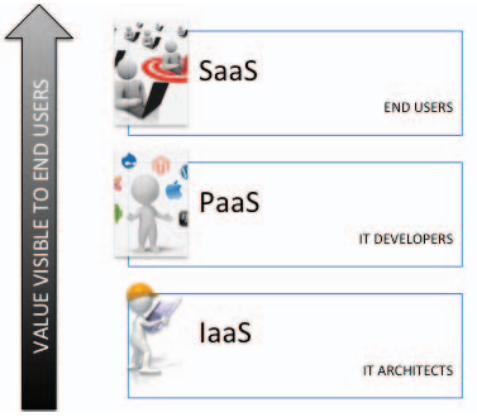

Figure 1. Cloud computing service models

In the last few years, several big companies were born or have refocused their business on CC: Amazon, VMware, Citrix, Google, Microsoft, Oracle, EMC2, NetApp, RackSpace, SalesForce just to mention a few. At the same time, a lot of service providers have developed different Cloud services, in term of both technologies and service models delivered.
Such a variety of offer simultaneously represents a huge opportunity for the end user and the cause of headaches mainly because of the heterogeneity of the services and the interfaces exposed by each service provider. The availability of a centralized service able to transparently grant a uniform access to different resources offered by different providers could represent the solution to the problem. Such a service should also grant the optimal use and the transparency of the available information among all the different providers accessible by the specific user [1].

Nowadays, there is a multitude of providers exhibiting technologies and services with similar functionalities, slightly differentiated in price and quality of service. The giant IT companies offer low-cost services, ensuring high reliability, resulting however in a chronicle lack of service customizations. A user who wants to either create or consolidate his/her IT infrastructures based on various Cloud services from various Cloud Providers is forced to look among different commercial solutions with different schemes relating to the costs of basic and additional services and to those related to the network traffic. Nowadays, there are no systems that allow brokering of CC systems to help users to compare different offers and to support the choice among them.

Cloud architecture is established as a standard for services provisioning. Discovering of services is a hard task since several services have to be analysed. The works [2] proposes a methodology for validating services requirements in a general framework for searching and interfacing services from multiple vendors in a cloud architecture, while [3] proposes a similar methodology for validating complex systems in general.

The transparent and effective resource brokering among different Cloud providers can be achieved by means of a Cloud federation platform acting as a normalization layer enabling the interoperability of different "traditional" CC platforms [4]. Within such a logical framework, a Cloud provider is able to augment its available resources by federating with other providers offering them, at the same time, its unused capabilities, thus gaining a profit by joining a federation [5].
In this paper, we describe a model that allows the federation of a wide variety of CC Platforms. Our "Cloud Orchestration and Federation" platform transparently federates resources from different CC providers and is based on an Orchestrator that represents the junction point between the user requests and the available resources; allowing the dynamic provisioning of the resources which better satisfy both the user requirements and the platform constraints in terms of general values as determined by appropriately defining a set of specific variables. The end-user is thus able to access a wide variety of resources by means of a single interface and in a completely transparent way; furthermore, the resources are dynamically instantiated in order to optimize predefined cost functions based on specific user-defined sets of parameters.
The remaining Sections of the paper are structured as follows:

- in Section II we report the state of the art of the technologies useful for our purposes;
- in Section III we describe the overall design of the Cloud federation platform;
- Section IV is devoted to the description of the prototype;
- in Section V we show the experimental results of the proof of concept we have developed;
- Section VI contains the conclusions.

II. State of the Art

In the history of Computer Science, the need for computational resources by organizations has basically followed three steps:

1. in the 70s, since information technology had an excessive cost, larger companies bought Mainframe which allowed to perform the processing of all user requests;
2. in 2000s the decrease in the cost of hardware and the ability to use different computers in a coordinated way, has encouraged the diffusion of GRID Computing systems in terms of standards for scientific elaborations;
3. since 2008 the increase in computing power has allowed the ability to emulate the hardware by means of specific software thus implementing the concept of virtualization.

CC represents the state of the art of the available technologies. It is strictly connected to the concept of virtualization including not only computational resources but also storage and network ones, all made completely automated for the system administrators and completely transparent for the end user.
Nowadays, there is a multitude of heterogeneous distributed DataCenter managers with different computing capabilities and different levels of compliance with the Cloud model. For example:

- virtualized DataCenter managers for supporting IaaS products (OpenStack, CloudStack, vCloud Director etc);
- virtualized DataCenter managers for supporting PaaS products (Cloudify, OpenShift, vCloud Director etc).

There are several platforms that allow the management of several Cloud Computing platforms using orchestration although they are intended to be used in organizations to manage internal resources. The dominant products in this area are:

- **BMC Cloud Lifecycle Management** is an hybrid cloud infrastructure management platform. BMC CLM supports multiple virtualization platforms such as XenServer, VMware vSphere, VMware vCloud Director, OpenStack, IBM pSeries, Microsoft Hyper-V. This platform is able to handle the activation request, the activation step, the monitoring and the dismantling of a Cloud platform. The use of the multitenancy features is ensured by the platform itself. This platform also allows to activate Databases instances on request (DBaaS) [6];
- **Parallels Automation for Cloud Infrastructure (PACI)** allows to create a virtual DataCenter based on containers and virtual instances. The system consists of different modules for task automation, billing, resource allocation and an online market [7];
- the **Zimory Cloud Suite** is a Cloud management platform that can create, expand, integrate and edit a Cloud infrastructure with platforms of different providers. The suite works in a centralized way, providing a single interface for managing IaaS and PaaS platforms through specific Application Programming Interfaces (APIs) [8];
- **Puppet** is a software for the management of Public Cloud infrastructures. It enables the construction of infrastructures in an automated way. The system reduces errors and minimizes the time of unavailability [9];
- the **Cloud Management Scalr Platform** allows the interaction with the main IaaS and PaaS platforms on the market and has an agent installed in the IaaS instances to automatically manage the scalability of resources [10];
- the **Abiquo platform** provides the management and control of multiple cloud platforms by using a unique user interface; it also allows developing applications to run on Cloud Public and the migration of applications to run on SaaS Cloud Public [11];
- the Cloud Management suite of **RightScale** enables the migration of applications between different infrastructures. The platform allows to check and manage resources and costs, to enable collaboration between developers and users, to emphasize and monitor the development lifecycle of the application, to automate the application deployment on Cloud infrastructure and to satisfy the SLAs required by the user [12].

III. System Architecture

"Cloud Federation" is a platform able to transparently and dynamically federate a wide variety of CC Platforms. An advanced Orchestrator module represents the connecting point between the user requests and the available and compatible resources, allowing the dynamic provisioning of the resources that better satisfy both the user and the provider requirements. The end-user is thus able to access a wide variety of resources by means of a single interface and in a completely transparent way; furthermore, the resources are dynamically instantiated in order to optimize predefined cost functions based on specific user-defined sets of parameters.

The ground-breaking nature of the system is represented by the possibility, for a generic end-user, to transparently access potentially any computational resource (CPU, storage, network) he/she needs, freely moving from one provider to another.

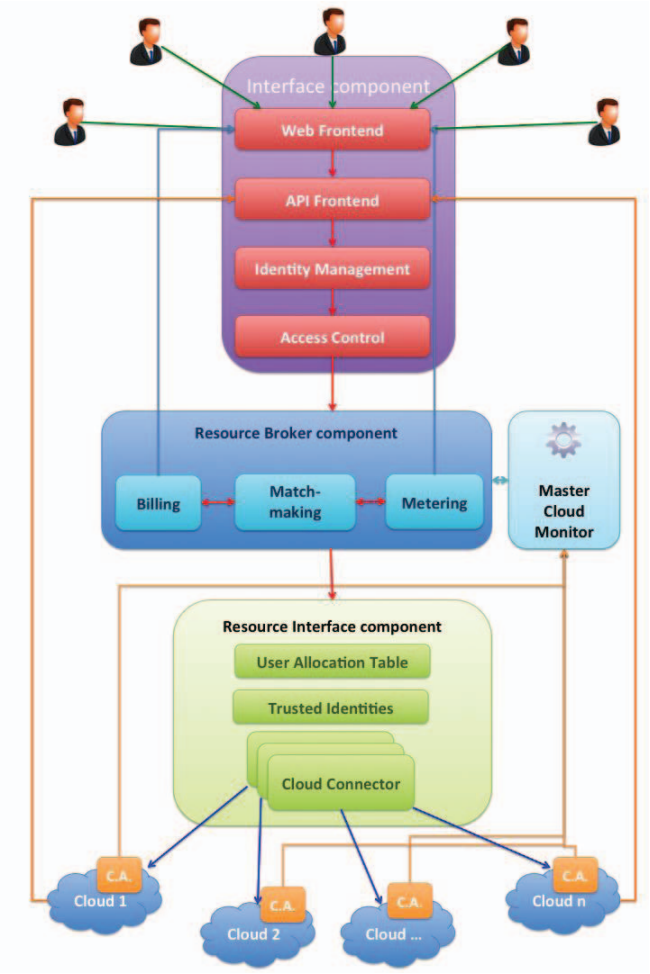

Figure 2. Cloud Federation architecture

The Cloud Federation platform achieves this goal by means of an architectural model based on three basic building blocks schematically shown in Figure 2:

- a Frontend component, representing the entry point to the entire platform;

- a Resource Broker, the very heart of the platform, consisting of Billing and Metering modules supporting an Orchestrator Engine module which dynamically matches the user requirements with the available resources;
- a Cloud Interface which represents the end-point connection to the federated resources.

In what follows we describe each of these components in detail.

A. Frontend module

The Frontend module allows the user to access the navigation features among the available services, to select the service and to handle the billing of the activated services. It represents the entry-point to the federation platform and consists of:

- a Web Frontend, offering basic interface services for individual users;
- an API Frontend, offering a machine-enabled interface for automated operations;
- an Identity Management (IM) module for controlling the user credentials.

Cloud platforms able to join a federation, can use the platform itself as an extension, interfacing to API Frontend, allowing to use other Cloud's resources transparently.

The API Frontend module is an intermediate level frontend accessible only by authorized software. The APIs provide access to all key functions of the system: search, selection and activation of a service for the end users and to platforms. Our idea is to develop it to be compliant with the Amazon EC2 communication protocol, also adopted by the OpenStack Nova module.

B. Broker module

Another basic element of the platform is the Resource Broker consisting of three modules:

- Metering module - it collects metrics from all the federated systems about accessibility and level of use of the resources, offered SLAs and specific information about running application;
- Billing module - it manages all cost data, also by using specific cost-tables for each provider, and calculates the service costs for the end-users, in agreement with the Metering Module;
- Match-Making module - it establishes the best fit between the user requests and the federated resource on the basis of a dedicated decision engine.

The platform allows end-user to experience a high quality of services also in terms of transparency and ease of access to resources. In particular, the Match-Making module allows the user to specify a generic resource description without any need for detailed knowledge of the accessible resources in the federated Cloud platforms, that can be heterogeneous. Furthermore, the expert system at the heart of the Orchestration Engine allows dynamic provisioning of resources, thus providing access to the resources available at a given time that best meet the user's requests.

The Match-Making module establishes the best fit between the user requests and the available federated resources on the basis of some specific decision engine. It has to manage the problem of dynamic allocation of the Cloud resources by using some strategies defining the best matching between available resources and users requirements. In our implementation, the Match-Making module activity is based upon the decision tree C4.5 algorithm [13]. This algorithm is supported by a graph of decisions which, if suitably crossed, allows to reach the node which identifies the best offer able to satisfy the user's request.

C. Resource Interface module

The Resource Interface module implements a system for collecting metrics, exchanging billing information and activating specific APIs to control the purchasing or dismissing of resources by the end-users. The module is able to dynamically interface with a wide variety of CC Platforms using the specific APIs made available by the providers.

The Resource Connector actually consists of several specific connectors to interface with resources provided by each of the federated infrastructure platform (eg. OpenStack, Cloudify, VMware etc.).

The Master Cloud Monitor (MCM) allows collecting Cloud Agent data to enhance Cloud interoperability. Cloud Agents are probes, installed in the federated Clouds able to read metrics in much detail than with the Resource Interface module.

IV. PROTOTYPE DEVELOPMENT

A prototype of the "Cloud Federation" platform has been implemented as a part of a collaboration between the University of Palermo and Telecom Italia. The prototype is running on a private Cloud and federates two OpenStack and one CloudStack platforms. The developed system basically represents a subset of the complete Cloud Federation architecture shown in Fig. 2. In particular, all the modules are implemented except for: the API Frontend, the Master Cloud Monitor and the Cloud Agent modules. The Billing and Metering modules have been implemented just in terms of basic features, since the prototype is meant to be a proof of concept.

A. Interface component

The Web Frontend module in based on AngularJS [14] and Twitter Bootstrap libraries [15]. AngularJS allows to implement the logic of web pages by using JavaScript on the client-side. Twitter Bootstrap is a valid solution to create websites accessible by a variety of devices on the market. The library consists of a CSS3 compliant stylesheet, a collection of icons and a JavaScript file that transforms the elements on the basis of the applied style. The developed prototype allows the user to select the specific service (a particular IaaS instance) based on the following criteria:

- the number of virtual cores;
- the amount of RAM memory;
- the size of virtual disk;

- the maximum price per hour required for running the instance.

The communication between the Frontend module and the Broker module is implemented via SOAP Web Services. These web services allow the decoupling of the interface module from the brokering engine and the federated systems. The information is exchanged in JSON format.

The Identity Management and Access Control modules are lightweight components developed by using Java Web Services and stores data on a MySQL database. Credentials are stored on a table and the web navigation is secured via a token control mechanism. The system refers to two privilege levels: an administrator level to set Cloud specific parameters and a user level to perform search and request of resources.

B. Resource Broker component

The Broker module is the orchestrator of user requests to the federated systems. The implementation details are described below:

1) *Match Making (MM) componen*t: it is the core of the platform. The activity of this module is based upon a statistical classifier that, in turn, is based on a decision tree built on the C4.5 algorithm; in particular we use J48 [16], an open source Java implementation of the C4.5 algorithm in the Weka data mining tool. The MM module builds decision trees starting from a set of training data based on the templates available on the federated Clouds. This module is supported by a graph of decisions that allows to reach the node which identifies the best offer able to satisfy the service request for the user;
2) *Billing component*: this component, based on Java Web Services and MySQL database, allows the system to calculate the price of the resources requested by the user taking into account the costs provided by the federated Clouds. In addition, it can generate bills for the user;
3) *Metering component*: this component is devoted to the collection of all the metrics available from the providers, thus allowing the system to evaluate the amount of resources available by each provider. It is developed using JClouds library.

C. Resource Interface module

The Resource Interface (RI) module represents the interface between the federation system and the federated provider. The system is able to dynamically interface with a wide variety of Cloud Computing Platforms. The core function of the RI module is to implement the real connection of the platform with the federated providers, each with its own communication protocol, for controlling resource activation, collecting metrics and indicators. In our prototype it is implemented by using Apache jClouds APIs [17]. Below we describe the details of each component:

1) *Trusted Identities component*: this module manages the credentials, encrypted stored in the database, that the platform has to use in order to authenticate with the federated platforms;
2) *User Allocation Table component*: the module stores in the MySQL database the user credentials and the association between users and activated services. It communicates via web services with the billing module in order to to estimate service costs to generate bills;
3) *Cloud Connector component:* this module, developed with the jClouds Java library, acts as the interface with federated systems. The library has been chosen for the vast amount of supported platforms. The component abstracts from any specific platform interface and allows to start an instance with basically any IaaS provider.

V. EXPERIMENTAL RESULTS

The testing field of the Cloud Federation platform consists of two OpenStack (one in Palermo and one in Turin) and one CloudStack (in Turin) instances. The entire platform allows a user to access a wide variety of resources by means of a single interface to select the best service in terms of both technical requirements and price policy. Furthermore, our system transparently and dynamically instantiates the assigned resources. In our tests, we have introduced virtual templates by defining specific sizes for RAM, disk, number of cores and relative price and costs. All these parameters are configurable as shown in Table I.

TABLE I. LIST OF TECHNICAL REQUIREMENTS THAT ALLOW A USER TO SELECT BEST SERVICE IN TERMS OF TECHNICAL REQUIREMENTS AND PRICE POLICY

Parameter	***Description***
CPU	Number of virtual CPUs presented to the instance.
RAM	Virtual machine memory in megabytes.
ROOT_DISK	Virtual root disk size in gigabytes.

Table II shows the templates used for the trial implemented within the project.

TABLE II. LIST OF TEMPLATES USED IN THE FEDERATION OF OPENSTACK AND CLOUDSTACK CLOUD PROVIDERS

Template	***CPU***	***RAM (MB)***	***ROOT_DISK (GB)***
OPENSTACK.TINY	[1–2]	[512 - 1920]	[1 - 10]
OPENSTACK.SMALL	[2 - 4]	[2048 – 3968]	[20 – 30]
OPENSTACK.MEDIUM	[4 – 7]	[3048 – 7912]	[40 – 70]
OPENSTACK.LARGE	[8 – 15]	[6192 – 16176]	[80 – 150]
OPENSTACK.XLARGE	[16 -32]	[14384 - 32688]	[120 – 380]
CLOUDSTACK.TINY	[1 - 7]	[2048 – 6192]	[100 – 200]
CLOUDSTACK.MEDIUM	[9 - 15]	[16176 – 24112]	[250 – 350]
CLOUDSTACK.LARGE	[16 - 24]	[24112 – 32176]	[400 – 500]

The MM module builds his decision trees from a set of training data based on the templates listed in Table II. Figure 4 shows the decision tree generated in our experiments. We have observed that CPU and ROOT_DISK represent discriminating parameters to manage the problem of dynamic allocation of the Cloud resources.

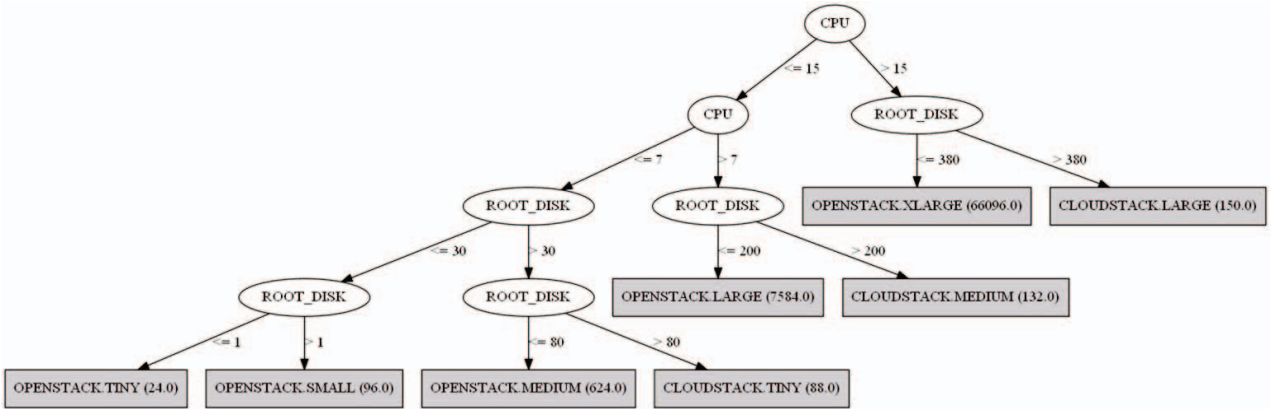

Figure 4. The decision tree generated by the Match-Making module using Openstack and Cloudstack templates.

Table III shows some of the searches performed by users on our platform with the resources dynamically and automatically instantiated.

TABLE III. SAMPLES OF SEARCHES PERFORMED BY USERS ON THE PLATFORM

CPU	*RAM (MB)*	*ROOT_DISK (GB)*	*PRICE*	*RESULT*
16	16000	240	250	OPENSTACK.XLARGE
16	16000	400	100	CLOUDSTACK.LARGE
8	16000	400	50	CLOUDSTACK.MEDIUM
8	16000	150	150	OPENSTACK.LARGE
4	8000	200	20	CLOUDSTACK. MEDIUM
4	8000	80	60	OPENSTACK.MEDIUM
4	8000	30	30	OPENSTACK.SMALL
1	2000	25	30	OPENSTACK.TINY
16	1600	250	50	NONE

Our tests confirm that RAM is not a discriminating parameters for the dynamic allocation of the Cloud resources. For more than 15 virtual cores (CPU=16) the best offer able to satisfy the user service request is dependent on the size of virtual disk. For virtual disk size less than 300 GB (ROOT_DISK=240) the provider proposed is *OPENSTACK* in its XLARGE version, while for disk size more than 300 GB (ROOT_DISK=400) the provider proposed is *CLOUDSTACK* in LARGE version. For less than 7 virtual cores (CPU=16) the virtual disk size is the only discriminating parameters. The providers proposed are *OPENSTACK,* in its MEDIUM (ROOT_DISK=80), SMALL (ROOT_DISK=30) and TINY (ROOT_DISK=25) versions and *CLOUDSTACK,* in its TINY version. The price actually represents a relevant constraint since, in some circumstances, inhibits the possibility to obtain a valid configuration.

VI. CONCLUSIONS

The described Cloud Federation prototype implements only the basic modules of the proposed complete platform. To implement an effective solution, robust billing and metering modules have to be developed. Furthermore, in a production scenario, the following aspects must be taken into account:

- lack of communication between the federation platform and the provider;
- dynamic change of the cost of resources;
- appropriate management of multi-tenant execution of instances;
- security issues of the federated provider that impose the exclusion from the federation.

The developed prototype is thus able to federate OpenStack and CloudStack platforms and to allow a customer to select and activate the best service in terms of technical requirements and price policy without to explicitly register to each of the federated providers.

ACKNOWLEDGMENT

This research has been supported by Telecom Italia and Università degli Studi di Palermo and funded by Telecom Italia within the “Cloud Federation” project.

REFERENCES

[1] M. Ambrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Soica., “Above The Clouds: A Brekeley view of Cloud Computing”. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-20, 2009.

[2] F. Moscato, F. Amato, A. Amato, “Model-driven engineering of cloud components in MetaMORP (h) OSY”, International Journal of Grid and Utility Computing, vol. 5, number 2, pp. 107-122, 2014.

[3] F. Moscato, V. Vittorini, F. Amato, A. Mazzeo, N. Mazzocca, “Solution workflows for model-based analysis of complex systems”, Automation Science and Engineering, IEEE Transactions on, vol. 9, number 1, pp. 83-95, 2012.

[4] P. Khanna, S. Jain, “Distributed Cloud Federation Brokerage: A Live Analysis”, 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing

[5] L. Mashayekhy, D. Grosu. “Cloud Federation in the Sky: Formation Game and Mechanism”. 2015. IEEE Transactions on Cloud Computing, Vol. 3, No. 1.

[6] http://www.bmc.com/it-solutions/cloud-lifecycle-management.html

[7] http://www.parallels.com/products/paci/

[8] http://www.zimory.com/index.php?id=259

[9] http://puppetlabs.com/solutions/cloud-management

[10] http://www.scalr.com/why-a-cmp/for-it-ops

[11] http://www.abiquo.com/service-providers/

[12] http://www.rightscale.com/cloud-portfolio-management/benefits

[13] http://en.wikipedia.org/wiki/C4.5_algorithm

[14] https://angularjs.org/

[15] http://getbootstrap.com/

[16] http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html

[17] https://jclouds.apache.org