DOI 10.12694/scpe.v16i1.1063
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 1, pp. 103–120. http://www.scpe.org

A TOOL FOR MANAGING THE X1.V1 PLATFORM ON THE CLOUD

EMANUEL MARZINI, PAOLO MORI*, SERGIO DI BONA, DAVIDE GUERRI, MARCO LETTERE†, AND LAURA RICCI‡

Abstract. The X1.V1 platform is a service oriented architecture, deployed on a set of Virtual Machines, that integrates services for managing fundamental activities in the ehealth scenario, such as Patient Identification, Clinical Document Repository, Prescription, and many others. The efficient provisioning of such services to citizens and to health professionals requires the adoption of increasingly powerful computing systems, in order to ensure that the growing number of service requests are served in acceptable time, even in case of computational peaks. Porting the X1.V1 platform on the Cloud could address the problem of the computational requirement mutability, since Cloud elasticity allows the reallocation of resources to Virtual Machines when necessary. This paper proposes a framework for managing the execution of the X1.V1 platform on the Cloud. This framework enables an easy, quick, and secure management of the Cloud resources allocation and reallocation to X1.V1 Virtual Machines, in order to enhance the platform performances, optimize resource utilization and, consequently, reduce the whole services cost. The design of the framework is focused on security aspects as well, because unauthorized accesses could lead to serious inefficiencies of the ehealth services. Finally, besides describing the architecture design and implementation of the X1.V1 Cloud manager framework, this paper also presents a set of experimental results which confirm the validity of the proposed approach to solve the X1.V1 platform performance issues in distinct reference scenarios.

Key words: Health records, Cloud computing

AMS subject classifications. 68M14, 97M60

1. Introduction. The spreading of the Electronic Health Record (EHR) is changing the vision of the European health systems, moving from a vertical approach to solutions able to organize and manage entire healthcare processes based on the integration of care settings. In order to fully support this view, these infrastructures need to be oriented to support care policies expressed as effective clinical services to the citizen/patient, both from a management and an optimization of resources standpoint. In fact, care and assistance needs are changing today as a consequence of the social, economic and technological advancements occurred in the last decade. Healthcare spending has strongly increased in all industrial countries, and in Europe it will potentially climb up to about 15% of the GDP (Gross Domestic Product) by 2020. This growth is largely driven by the increase in demand due to the average life expectancy and, as consequence, to the number of people affected by chronic diseases which becomes more expressed with increasing age. This new situation has three different but interconnected perspectives:

1. the clinical services managers are challenged to deliver effective healthcare services at reasonable costs;
2. the clinicians are interested to apply and improve the effectiveness of clinical guidelines, increasing the emphasis on evidence-based medicine;
3. the citizens are increasingly demanding high quality and continuity of care, they are more aware of clinical risk management and are claiming more efficient healthcare services, with reduction of waiting times and a better utilization of resources.

In an increasingly digital economy, in order to face these challenges, the healthcare cannot continue to be a service where the user is passive. This trend is reflected in EHR infrastructures which are becoming increasingly arranged and enriched so that the citizen/patient can be more involved in her/his own care, gradually moving away from a passive role. At the same time, in order to guarantee continuity of care across different care settings it is necessary to provide healthcare operators with an integrated set of services and facilities for defining, monitoring, evaluating and fine-tuning personalized care programmes.

In order to make this transition technically and economically viable it is necessary to consider a definitive move towards new organizational models able to govern seamless integration across government, community and private information systems (EHRs, Personal Health Records PHRs, Telecare centres). Such models must be supported by distributed architectures where users, systems, applications, and devices are able to collaborate.

*Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, Italy (paolo.mori@iit.cnr.it).
†Dedalus SpA, Livorno, Italy (name.surname@dedalus.eu).
‡Dipartimento di Informatica, Università di Pisa, Pisa, Italy (laura.ricci@unipi.it).

Service oriented architectures (SOAs), based upon largely accepted standards, allow heterogeneous systems that support different care environments interacting and combining personal, social and environmental issues that affect the citizen/patient's healthcare processes.

This scenario will progressively push the healthcare organizations to adopt more and more powerful computing systems to cope with the increasing workload related to the management of the new ehealth services. Moreover, huge repositories are required for managing the very large and continuously increasing number of clinical documents. In addition, as a consequence of the availability of more and more complex services, the computational requirements can change significantly from one service to another, or they can even change during time. For example, the computational requirement of the prescription process, starting from the General Practitioner (GP) that prescribes a drug or a treatment up to the drug delivery or administration, is likely to be high during daytime, i.e., during working hours of the GPs, but most likely it is low during the night when only emergency doctors use this service. As a counter example, services related to audit and control processes are more likely to run during the night. By adopting Cloud-based paradigms, healthcare organizations as well as complex systems could receive significant benefits from several points of view. In particular, one of the essential characteristics of a Cloud system is the elasticity [14], that is defined by NIST as: "Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time" [20]. Cloud elasticity is a crucial feature in our scenario, since it can help healthcare organizations to cope with the mutability of the computational and storage requirement of the ehealth systems and services. With the main goal of validating the approach proposed in this paper, we have taken into account a commercial platform, distributed by Dedalus SpA[1] (an Italian company which is national leader in healthcare software), aimed at guaranteeing the interoperability among the different interacting actors of the healthcare scenario. The X1.V1 platform is a service oriented architecture, deployed on a set of Virtual Machines (VMs), that integrates services for managing fundamental activities in the ehealth scenario, such as Patient Identification, Clinical Document Repository, Operator Profiling, Access control and authorization, as well as Prescription and Reporting processes.

This paper presents a Cloud Management System, named X1.V1 Cloud Manager, which allows the execution and the management of the X1.V1 platform on a Cloud system through a simple graphic interface and is able to change the allocation of the Cloud resources to the X1.V1 services in just one click in order to promptly react to a computational peak, or according to a predefined scheduling to face a forthcoming expected overload. These activities, in fact cannot be performed manually since it would be difficult, time consuming and error prone for the system administrator. The X1.V1 Cloud manager is able to manage the VMs that composes the X1.V1 platform and to change the overall resources allocation following some predefined configurations. Each configuration represents the optimal allocation of the Cloud resources to obtain the best performances of the platform in a specific usage scenario. Switching from one configuration to another has a great impact on the X1.V1 platform performance, and switching to the wrong configuration could even result in a denial of service in some critical situations. Hence, the access to the X1.V1 Cloud Manager must be protected to avoid unauthorized and inappropriate reallocations of Cloud resources. In particular, at least two distinct access levels must be defined: one that allows the system administrator to select a subset of the existing configurations (e.g., the ones that do not involve additional costs for the customer), and another that allows the system administrator to switch to any configuration, and to upload customized configuration to face with unusual or emergency scenarios. Summarizing, the main feature of the X1.V1 Cloud manager is the easy exploitation of the Cloud elasticity, while providing adequate security to protect the processes of the ehealth services.

This paper is organized as follows: Section 2 gives a brief overview of the X1.V1 platform, Section 3 describes the main reasons for executing the X1.V1 on the Cloud, and Section 4 describes some related works. Section 5 describes the main features of the X1.V1 Cloud Manager, Section 6 gives some implementation details and describes a set of experiments, and Section 7 concludes the paper.

2. The X1.V1 platform. In order to support the interoperability among the different care environments (hospitals, primary care, homecare) and to provide the healthcare operators with mechanisms for implementing

[1] http://www.dedalus.eu/

integrated care pathways, starting from 2005 Dedalus defined, designed and implemented the interoperability platform, namely X1.V1 [18]. In particular, X1.V1 was designed to address the requirements of both national or regional healthcare service models, when the territorial organization is divided into self-regulatory administrative regions or municipalities. From a functional point of view, the X1.V1 platform is the enabling tool for implementing the EHR at regional and national level and the EPR (Electronic Patient Record), by supporting healthcare and clinical processes at enterprise level. Moreover X1.V1 could be used for supporting pathologys networks, second-opinion services and decision making processes with data collection and performance indicator elaborations. The X1.V1 platform is currently adopted by several Italian healthcare organizations, such as the Italian Regions Abruzzo, Umbria and Marche, for the EHR management. From a technological point of view, it implements a federative architecture, based on the standard IHE (Integrating Healthcare Enterprises) XDS (Cross Enterprises Document Sharing) integration profile. In particular, the X1.V1 platform follows a Service oriented approach by virtualizing the following components:

- Single-Sign-On (SSO) module, for managing authorization and authentication functions;
- Master Patient Index (MPI) module, for supporting the unique identification of patients among the different care environments;
- Master Code Index (MCI) module, for shared management of coding and terminologies;
- Document repositories, able to store all the managed clinical documents produced in the different care environments;
- Document Registry, able to store the relevant metadata of the documents and to provide the index of the documents stored in the repositories;
- Enterprise Service Bus (ESB) and a Workflow engine, for the orchestration of the clinical processes implemented through the platform.

The main information systems interoperating with the platform are the EMRs (Electronic Medical Records) of the General Practitioners, the Hospital Clinical Information Systems (HCIS), including EMRs, first aid and operating theatres applications, the Laboratory and Radiology Information systems, as well as non-clinical systems, such as booking systems, ADT. Moreover the platform is able to interoperate with the Pharmaceutical information systems and end-users/citizens viewing interfaces. The platform is able to support among the others the following activities:

- The eDocuments shared management, such as the publication and the accesses of the Patient Summary, the Patient Care Coordination Report, the Medical reports produced by the HCIS;
- the ePrescription processes;
- integrated care pathways related to predefined care models (such as Chronic Care model);
- the eBooking of health services;
- the events related to health episodes.

The use of the X1.V1 solution therefore needs the deployment and the management of a very complicated healthcare ecosystem, that requires an heterogeneous, time dependent and scalable allocation of resources.

3. Executing the X1.V1 Platform on the Cloud. The increasing popularity of Cloud systems [20] [30] is mainly due to the possibility to exploit (virtualized) resources when users actually need them for executing (parts of) their processes. Distinct Cloud service models have been defined [15], depending on the kind of resources that are provided. The framework proposed in this paper exploits the Infrastructure as a Service (IaaS) model, where the resources provided to users are computational infrastructures consisting of Virtual Machines (VMs) connected by virtual networks. When requesting VMs, users can choose the most proper network configuration, VM features (e.g., CPU count and memory space), VM images (i.e., the operating system they need for their application) and they can install and run on their VMs the applications they need. Once requested, the virtual computational infrastructure is available quickly and the number of machines and/or their features can be increased or decreased by the users on demand during the computation and according to their needs. Cloud users are charged by Cloud providers for resource usage, depending on the number of machines, on their features, on the computational time used, etc..

Public IaaS Cloud facilities are currently provided by many big companies such as Amazon[2] and Microsoft[3],

[2] http://aws.amazon.com/ec2/
[3] http://www.microsoft.com/windowsazure/features/compute/

and a number of frameworks, such as OpenNebula[4], Eucalyptus[5] or OpenStack[6], are available to deploy Cloud systems in users' data centres (Private Clouds) [31] [33].

The X1.V1 platform provides a set of services concerning several aspects of the ehealth scenario, such as Patient Identification, Clinical Document Repository, User Profiling, Prescription and others, and these services are already deployed on a set of VMs. Hence, executing the X1.V1 platform on the Cloud is straightforward. Allocating a static set of Cloud resources to each of these VMs could result in low performances as well as in a waste of resources, since the computational load of the services is variable over time. We report here two reference examples taken from the normal operation of the X1.V1 platform.

Example 1: The X1.V1 platform computes some statistics during the night, while the other services are likely to be unused. This is a very heavy task, and sometimes the results are not ready before the day after when daily services should start again. Hence, the load of the VMs involved in this task is very high during the night, while it is negligible during the day. Consequently, during the night hours the VMs that perform this task should have a large number of resources, while the VMs running the other services (such as the Booking or the Prescription one) should have a reduced number of resources.

Example 2: The Prescription service is used mainly by GPs, Pharmacists and Specialists to perform the ePrescription process. In fact, the GPs exploit the service to store the drugs and diagnostic exams prescriptions for their patients, and the Pharmacists and Specialists use the same service to retrieve such prescriptions and to update them according to drugs provided or exams executed. The Prescription service is likely to receive a large number of service requests during the working hours of GPs, Pharmacists and Specialists. In particular, there could be some computational peaks, i.e. a very large number of requests that are issued in a short period of time. Consequently, since each request is issued by a person that is waiting for the answer, in order to serve each request within a reasonable response time, the resources allocated to the VM(s) running the Prescription service should be increased during the working hours of GPs, Pharmacists and Specialists.

The previous examples point out that the usual behaviour of the X1.V1 platform from the point of view of resource requirements is known. In other words, during the daily operations there are predictable variations of the computational load of the X1.V1 VMs. Hence, the execution of the X1.V1 platform on a IaaS Cloud would greatly benefit from the Cloud elasticity in order to grant a given quality of service, i.e., to ensure an upper bound to response time of the provided services [4]. As a matter of fact, the framework could increase the resources assigned to a given VM, and hence to the services deployed on it, when a computational peak is expected for this VM. Symmetrically, the framework could reduce the resources assigned to other VMs when a low computational load is expected for these VMs. Performing the management and the update of the allocation of the Cloud resources to the X1.V1 VMs manually is not convenient, because it requires the knowledge of the allocation of the services to the VMs, it would be time consuming and error prone for the system administrator. For this reason, this paper proposes a framework for the management of the allocation of the Cloud resources to the VMs that compose the X1.V1 platform.

Besides the benefits previously described, the adoption of Cloud computing to perform critical tasks introduces also security issues. These security issues are described by the European Network and Information Security Agency (ENISA) in the report "Cloud Computing. Benefits, Risks and Recommendations for Information Security" [10], and by the Cloud Security Alliance (CSA) in the reports "The Notorius Nine. Cloud Computing Top Threats in 2013" [8] and "Security Guidance for Critical Areas of Focus in Cloud Computing" [7].

4. Related Work. A number of works in the literature, such as the ones in [2], [6], [16], and [25] describe some approaches for the design and implementation of several health related services (such as the EHR or health monitoring systems) on the Cloud. Instead, the focus of this paper is different from these works because we take for granted that the X1.V1 platform is already designed for being executed on the Cloud, and we focus on the design, implementation and validation of a tool which eases the exploitation of the Cloud elasticity, in order to allow the X1.V1 platform to take maximum benefit from the Cloud environment.

The authors of [3], instead, propose a tool, called Elastack, which is an automated monitoring and adaptive system, generic enough to be applied to existing IaaS frameworks, and intended to provide elasticity to these

[4]http://opennebula.org
[5]http://www.eucalyptus.com
[6]http://www.openstack.org

frameworks. As a matter of fact, the authors state that the elasticity support of the currently available Cloud frameworks is quite immature, and they define their own tool to support elasticity in IaaS Clouds. Their framework is based on the Serpentine middleware [19]. Elastack reacts to workload peaks by creating new instances of VMs, and these instances receive some tasks to be performed. These new instances are then destroyed when they are no longer necessary.

Roy et al. [26] describe a framework for efficient autoscaling in the Cloud, aiming at guarantee a given Quality of Service (QoS) level. Their framework exploits a look-ahead resource allocation algorithm based on model-predictive techniques for workload forecasting which allocates and deallocates VMs to the application trying to optimize the amount of used resources while respecting the promised QoS and minimizing the operational cost.

The approach proposed in [1] defines two adaptive controllers which use both reactive and proactive controls to dynamically change the number of VMs allocated to a service based on the current and on the predicted future load of that service.

The previous approaches differ from the one proposed in this paper because they resolve performance issues simply by allocating new VMs or deallocating existing ones, and this requires that multiple instances of the applications deployed on the VMs can be run in parallel. The X1.V1 Cloud Manager, instead, does not simply increase the number of the VMs of the system to react to a computation peak. In fact, the X1.V1 Cloud Manager defines a set of configurations exploiting at best the Cloud resources assigned to the X1.V1 VMs in some specific scenarios (such as the ones described in the two reference examples in Section 3). Hence, the X1.V1 Cloud Manager could resolve a temporary performance issue due to the overloading of some services even without increasing the total amount of resources assigned to the X1.V1 platform, but simply choosing a different allocation of the same resources to the VMs of the X1.V1 framework.

CloudScale [29] is a prediction driven system that automates fine-grained elastic resource scaling for Cloud infrastructures. CloudScale does not assume any prior knowledge about the applications that are running on the Cloud, and it exploits an online resource demand predictor, described in [13], that is application agnostic and that achieves a good prediction accuracy for a range of real world applications. CloudScale also defines two complementary handling schemes to be executed when the resource demand predictor underestimates the resource requirement, to update the resource allocation thus avoiding the violation of the previously defined service level objectives.

The main difference between the X1.V1 Cloud Manager and the approaches proposed in [26], [1], and [29] is that they address two different problems: the X1.V1 Cloud Manager assumes a deep knowledge of the application that is running on the Cloud, the X1.V1 platform (because the configurations that describe the resource allocation are defined according to the usual application behaviour), while the other approaches do not assume any knowledge of the applications that are running on the Cloud, and for this reason they are focused on predicting the future resources requirement of these applications in order to promptly update the resources allocation.

CloudScale[7] is also a research project funded under the European FP7 programme. The main aim of the project, introduced in [5], is to provide an engineering approach for building scalable cloud applications and services based on the state of the art cloud technologies and software. The CloudScale approach will make cloud systems scalable by design, i.e. they will fully exploit the cloud elasticity during their operations to provide scalability, while minimizing the amount of computational resources used. One of the reference scenario addressed by CloudScale will be the EHR one. Since [5] is a work-in-progress paper, it simply provides an overall description of the Cloudscale approach, and it does not allow us to make a comparison with our approach.

A set of further works concerning other approaches to exploit Cloud elasticity are described in [12].

Finally, the difference between the X1.V1 Cloud Manager and the graphical tools provided by the main Cloud frameworks (e.g., Sunstone and Horizon) is that the X1.V1 Cloud Manager allows to transparently manage a number of distinct Cloud providers, which hosts the X1.V1 VMs, and the configurations are customized for the computational requirements of the X1.V1 framework in several scenarios.

5. X1.V1 Cloud Manager. The X1.V1 Cloud manager is a framework meant to enable an easy, quick, and secure management of the resources allocated to the X1.V1 platform in the Cloud environment, in order to

[7] www.cloudscale-project.eu

enhance the platform performances, optimize resource utilization and, consequently, to reduce the whole cost of the computation. In particular, the main aim of the X1.V1 Cloud Manager is to enable the system administrator to easily deploy and manage the VMs that build up the X1.V1 platform in the Cloud environment, in order to quickly react to computational peaks by properly updating the resources allocation to the X1.V1 VMs. In fact, the X1.V1 Cloud Manager allows the system administrator to define a set of configurations, where each configuration defines in details the resources to be allocated to each VM of the system (e.g., the number of virtual CPUs or the size of the RAM memory) and other configuration details (e.g., the Cloud Provider on which each VM will be deployed). In general, each configuration defines the best resource allocation to obtain a good performance in a specific scenario, such as the GPs' labs opening hours one. With reference to the previous example, the system administrator could define the following configurations:

Normal: allocates the same amount of resources to all the VMs;

Interactive: allocates more resources to the VMs hosting the interactive services, i.e., the services which require interactions with people (e.g., doctors, patients), such as the Prescription service, reducing the resources allocated to the statistic service and or to other services;

Night: allocates a small amount of resources to the VMs that implement the interactive services, and reserves high computational power and large memory space for the VMs responsible to perform statistical analysis or data warehousing on the available data;

Power: allocates high computational power and a large memory space to all the VMs of the X1.V1 platform.

Many other configurations could be defined by the system administrator, to handle other possible situations where a different allocation of the resources to the X1.V1 VMs could lead to better performances.

The X1.V1 Cloud Manager allows the management of distinct Cloud providers at the same time, even running distinct Cloud softwares, such as OpenStack or OpenNebula. Hence, a configuration file could state that some VMs of the X1.V1 platform can be deployed on a provider running OpenNebula and other VMs on another one running OpenStack. Switching from a configuration to another, besides changing the allocation of the resources to the VMs, could even migrate some of them from a Cloud provider to another.

Depending on the Cloud provider and on the related agreement, some configurations could result in additional costs. As an example, we could define the *Normal*, the *Interactive*, and the *Night* configurations in a way such that they require the same amount of resources, i.e., they define three different ways of partitioning these resources, while the *Power* configuration requires more resources than the others. We also suppose that the *Power* configuration uses more resources than the ones agreed with the Cloud provider (or available in the user data center in case of Private Cloud); hence the *Power* configuration could lead to additional cost for the Cloud user. Hence, in order to reduce the cost of the computation, during the normal operation of the X1.V1 platform the first three configurations should be used, while the *Power* configuration should be exploited only when it is really necessary to guarantee a good service level. The X1.V1 Cloud manager represents the configurations previously described through a OVF [9] file (with extensions).

When the system administrator selects a configuration, the X1.V1 Cloud Manager performs all the operations required to update the VMs according to the new configuration, such as changing the amount of memory or the number of virtual CPUs allocated to the VMs or migrating the VMs from one Cloud Provider to another.

The X1.V1 Cloud Manager includes a monitor component that collects information from the running VMs, such as the CPU workload, the amount of memory used and the network traffic. This information is elaborated in order to provide aggregated data that could help the system administrator to decide whether the current configuration is resulting in good performance or a more suitable configuration can be chosen.

We plan, as a future work, to provide the X1.V1 Cloud Manager of a further component which automatically decides whether, in a given scenario, a new configuration should be adopted, i.e., whether another resource allocation would result in better performance of the X1.V1 platform. This component will include a resource demand predictor that will exploit the information collected by the monitor system in order to perform the configuration switch before the actual change of the load of the X1.V1 VMs and only if the time spent to perform the configuration switch is paid by better performance.

The X1.V1 Cloud Manager has been designed to provide an adequate security level for the processes and data involved in the X1.V1 platform. In particular, this paper is focused on the security of the X1.V1 Cloud Manager, in order to prevent unauthorized users to change the current configuration. As a matter of fact,

adopting the wrong configuration could lead to a significant degradation of the performance of some X1.V1 services, thus causing a denial of service. Instead, this paper does not cover the security issues specific of the X1.V1 platform because it assumes that the X1.V1 platform already provides an adequate level of security.

The security support of the X1.V1 Cloud Manager includes the authentication and the authorization phase. The authentication phase exploits the LDAP certificate-based authentication method [28], [32], which provides an adequate security level for the critical applications such as the X1.V1 platform. Each entry of the LDAP directory represents a X1.V1 administrator, i.e., a subject who has some rights concerning the management of the X1.V1 platform on the Cloud, and include a set of attributes that are paired with the corresponding X1.V1 administrator, that will be used in the authorization phase for evaluating the access control policy. The *role* is an example such attributes. Other attributes (static or even dynamic, i.e., that change over time) could be introduced in order to define complex access control policies. The authorization component is aimed at checking that the user that authenticated herself/himself on the Dashboard really holds the right to perform the operation she/he requests, because distinct sets of rights are assigned to distinct (classes of) users. To design both the architecture of the authorization system and the security policy language, we rely on the XACML standard [23] developed by the OASIS consortium, because it is well known, widespread, and supported both by academic (and, consequently, free and open source) such as WSO2 Balana, and by commercial tools, such as the Axiomatics authorization framework [11]. The XACML standard naturally allows to implement the Role Based Access Control (RBAC) model [27], [22], i.e., to define a set of roles and to pair a set of rights to each of the roles. In our scenario we could have (at least) two roles, ADMIN and SUPERADMIN. The role ADMIN will have the right to select a subset of the existing configurations only, i.e., the ones that do not require to pay an additional fee to the Cloud Providers, while the role SUPERADMIN will have the right to select any of the existing configurations and even to load a new and customized configuration written for the specific situation she/he is coping with. For what concerns the interactions with the Cloud Providers, instead, we exploit at best the security mechanisms they provide for the authentication on the Cloud platform and for securing the communications between the X1.V1 Cloud Manager and the Cloud platform.

5.1. Architecture. The architecture of X1.V1 Cloud Manager is shown in Figure 5.1, and consists of the following components.

Web Dashboard. It is a web application providing a graphical interface which allows to easily manage the configuration of the cloud services that build up the X1.V1 platform. All the configurations stored in the repository are listed in the dashboard, and the administrators can switch from the current configuration to another one in just one click, i.e., simply pushing the related button, provided that they hold the proper right. Authorized administrators could also upload customized configurations to deal with special or emergency situations. The Switch Configuration Command (SCC) is sent to the Configuration Engine. This interface also allows to monitor the resource usage of the VMs of the X1.V1 cloud system, properly represented through a set of graphs. Figure 5.2 shows the dashboard interface that allows the configuration switch, while Figure 5.3 shows the dashboard page that represents the resource usage of each VM.

Configuration Engine. This component is the core of X1.V1 Cloud Manager. This module receives the Switch Configuration Commands (SCC) from the dashboard, processes and executes them on the Cloud Providers that host the final services. Each SCC requires to change the Cloud resource allocation according to the related configuration file. The directives listed in the configuration file need to be mapped into a set of specific commands to be sent to the Cloud Providers to perform the required resource reallocation. For this reason, a specific driver for each Cloud Provider has been included in the architecture, to translate the SCC into a set of commands that can be executed by the Cloud Provider. Finally, the Configuration Engine embeds the Policy Enforcement Point (PEP) that triggers the authorization process.

Security Manager. It provides a cross support for the authentication and authorization of users accessing the X1.V1 Cloud Manager. The authentication system is based on a LDAP server that is invoked by the web dashboard. After a successful login, the authentication system also returns a set of user attributes, such as the user ID and Role, that are necessary to perform the authorization process. The authorization process is triggered by the Policy Enforcement Point embedded in the Configuration Engine, which invokes the authorization system to check the authorization policies every time it receives a SCC. The access control policies are written in XACML language, a widely used standard developed by OASIS consortium. In the reference scenario, the policy exploits

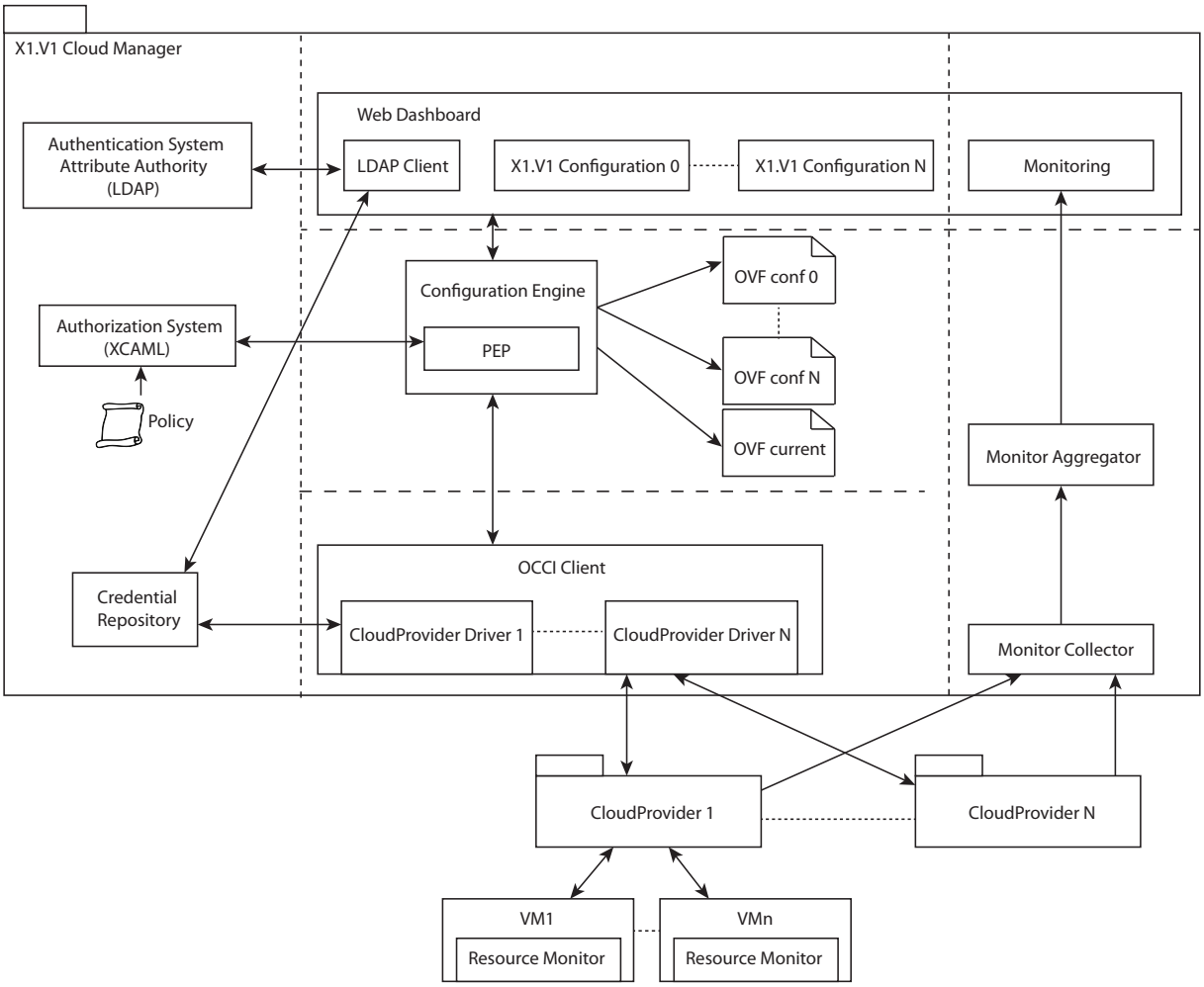

FIG. 5.1. *Framework Architecture*

the Role Based Access Control (RBAC) model, where a set of roles are defined and a set of rights is paired to each of these roles. One (or more) role is assigned to each user, thus determining the users rights. However, the XACML language is very expressive, and allows to write more complex security policies than the pure RBAC one. As a matter of fact, besides the role, other attributes can be stored in the LDAP server and exploited in the access control policy for the authorization decision process.

Driver for Cloud providers. The drivers are in charge of the communications with the Cloud Providers, i.e., of customizing the commands defined by the Configuration Engine for the specific Cloud Provider these commands are to be sent. Hence, a specific driver must be instantiated for each distinct Cloud provider exploited for the execution of the X1.V1 VMs. In general, a driver could also be instantiated to exploit a Public Cloud, such as Amazon EC2, for the execution of some of the VMs. The drivers also manage the channels used for the request/response to the Cloud Providers and the channels used for the monitoring communications (data and alert). The channel for the request/response is a simple OCCI interface common to all the Cloud drivers present in the system. The drivers also manage the security of the interactions with the Cloud Providers. In particular, a set of local users is statically defined on each Cloud Provider, and each driver exploits the right user to interact with each Cloud Provider. Secure communications must be exploited with the Cloud Providers that support them.

Monitoring System. For each of the VMs running on the Cloud Providers, the monitoring system collects a set of usage parameters, such as the CPU and RAM utilization and network statistics. The monitoring system also allows to set some alarms when a metric exceeds a threshold. The alarms have two level of emergency: WARNING and FAILURE. When a VM notifies the violation of a threshold the system forwards the notification to the dashboard, in a way such that the administrator can change the current configuration. The monitoring information is sent from the Cloud Provider to a collector module present in the Cloud Manager. This module collects both the statistics and the alarms coming from the Cloud Provider, and it passes these data to a Monitoring aggregator that splits the hardware statistics and the alarms. Finally, the monitoring information are displayed in the monitoring section of the Web dashboard in a proper graphical format, that also provides a clear visualization of the alarms (Figure 5.3).

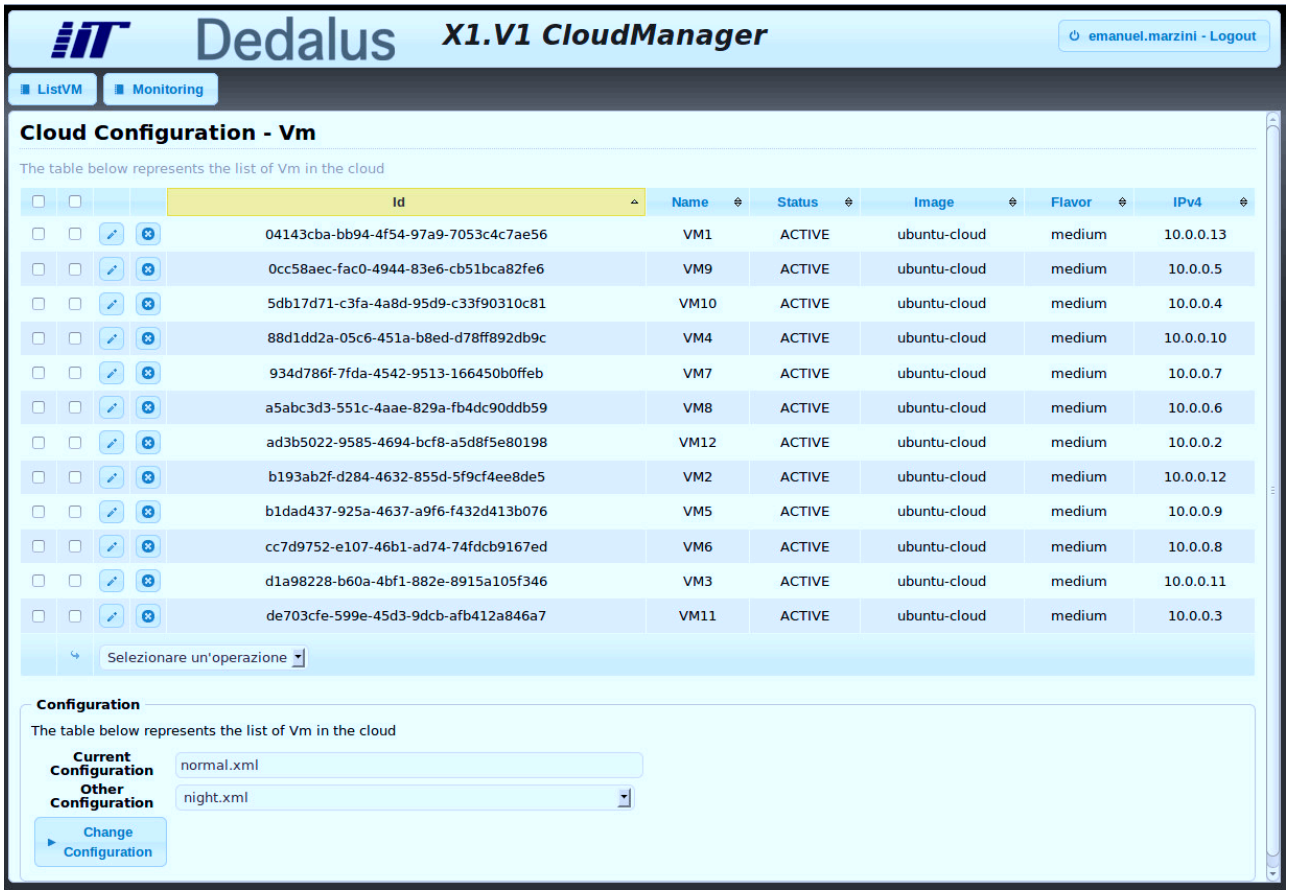

FIG. 5.2. *Dashboard Configuration Switch page*

6. Prototype. This section describes the main features of the prototype we developed, that implements most of the functionalities of the architecture described above, and presents a set of experimental results.

6.1. Implementation Details. The X1.V1 Cloud Manager core is mainly developed in Java EE under Apache Tomcat web server. The Web dashboard uses the typical Java EE tools like a JSP and Servlet. Ajax and the Highcharts library have been used to display graphs.

The authentication system exploits the certificate-based authentication method provided by the LDAP protocol. In particular, we exploited LDAP over SSL (LDAPS), which allows to use a secure channel to communicate the credential in the authentication phase instead of a plain one. We added custom attributes in posixAccount to retrieve credentials to communicate with OpenStack. Once retrieved, these values are used

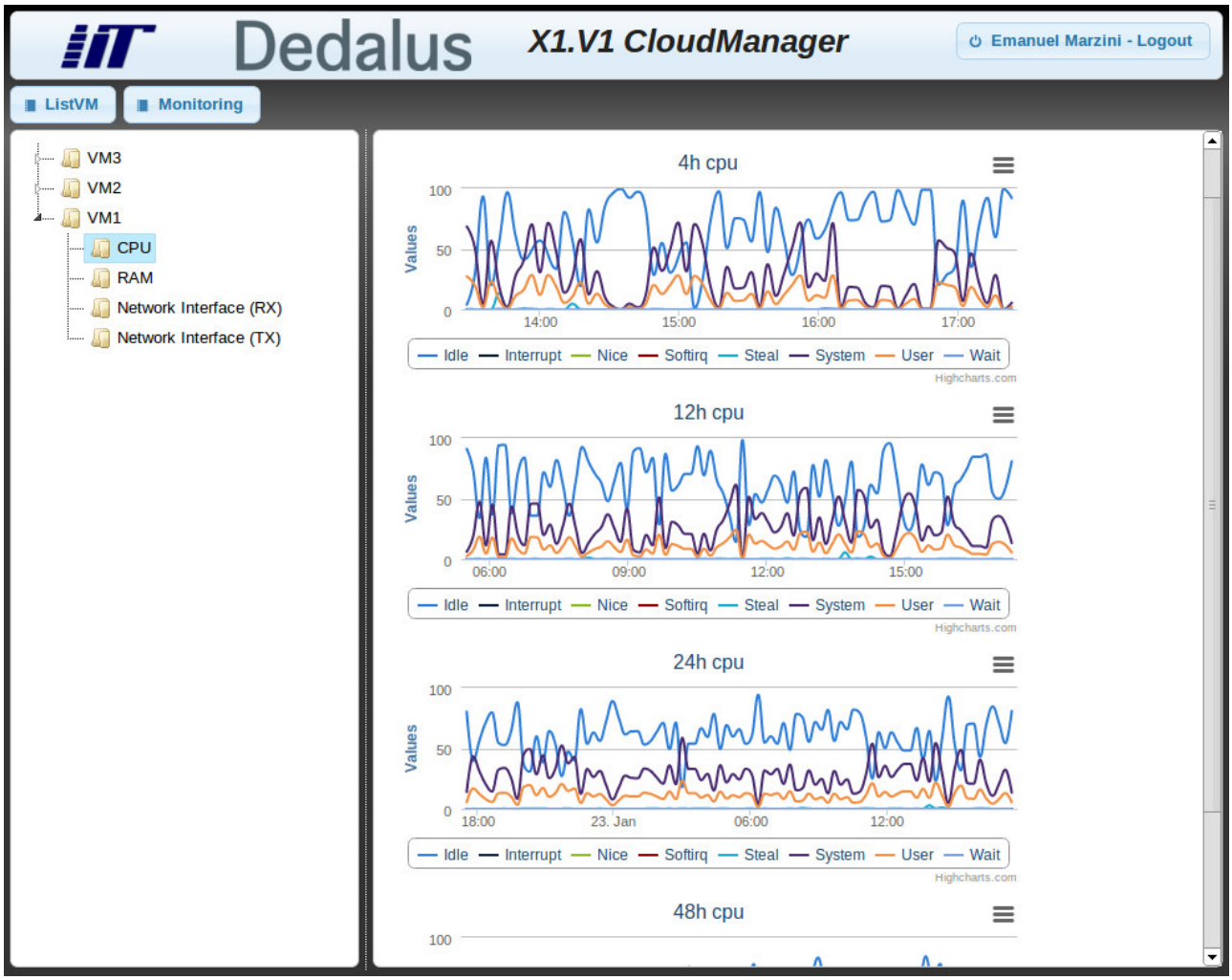

FIG. 5.3. *Dashboard Monitoring page*

once for the initial authentication on OpenStack, and they are not saved anywhere. Our implementation expoits the OpenLDAP[8] software to set up both the LDAP server and the client within the dashboard.

The authorization system is based on the WSO2 Balana[9] XACML engine, which is an open source software for evaluating access control policies written in XACML v3.0. An alternative XACML engine that could be adopted is the one released by SUN[10]. However, SUN's engine supports XACML v2.0 only, it is no longer maintained and it is considerably slower than the WSO2 Balana engine. Another XACML engine is the one developed by Axiomatics, which is a commercial product[11]. In particular, a significant advantage is that Axiomatics provides an authorization framework covering all the phases of the authorization process, such as policy writing, enforcing and tuning.

The communications among the Configuration Engine and the XACML authorization system follows the SAML protocol [21] and the opensaml library is exploited.

The current version of our prototype implements one Cloud Provider driver only, i.e., the OpenStack one,

[8]http://www.openldap.org/
[9]http://xacmlinfo.org/category/balana/
[10]http://sunxacml.sourceforge.net
[11]http://www.axiomatics.com/solutions/products.html

because OpenStack is a widely used open source software for setting up private Clouds, and it is the one we used for our experiments. OpenStack does not require to shutdown the VMs to change their resource allocation. Hence, the X1.V1 services are simply suspended when a configuration switch is performed, and they are available again right after the resource reallocation. For the Cloud Provider installation we chose the OpenStack Havana release with the fundamental modules installed on two distinct nodes: the Controller Node and the Compute Nodes. The Controller Node hosts Keystone (Security manager), Glance (Image manager) and Nova (VM manager), and it manages all the request from the X1.V1 Cloud Manager. The Compute Nodes host only the compute agent for Nova and the network module. These nodes receive the requests from the Controller Node and manage the execution of VMs. In our prototype the network communications are managed through nova-network without the specific module Neutron.

As far as concerns the authentication on the Cloud providers, OpenStack provides a token system to allow interactions through its API calls. When the user successfully performs the initial authentication phase, Keystone assigns a temporary token, and this token can be used by the OpenStack driver to send commands to the Cloud provider. The token expiration is setted to 24h, but this parameter is configurable by OpenStack. The initial authentication on each Cloud provider is executed when the user performs the authentication on the X1.V1 Cloud Manager. In particular, a PKI infrastructure is used to authenticate on OpenStack, and the required key is stored in a password protected Keystore, that is opened by the user at authentication time. This credential allows the OpenStack driver to perform the initial authentication phase on the Cloud provider and to obtain the authentication token required to invoke the subsequent commands to manage the Cloud resource allocation. For this reason we have a secure channel to communicate among X1.V1 Cloud Manager and OpenStack Cloud provider.

The monitoring system exploits Collectd[12] to retrieve the usage information from the VMs. There is a client on each VM in the cloud that retrieve the statistics of CPU usage and RAM utilization. These data are sent to a collector daemon in each cluster that redirect the statistics to X1.V1 Cloud Manager. The data collected are analyzed and visualized on the web dashboard.

6.2. Experimental Results. In order to validate the proposed framework, we performed a set of experiments to evaluate the time required to switch from one configuration to another varying the number of VMs in the system. In particular, we measure the time from the moment when we press the switch configuration button on the X1.V1 Cloud Manager interface to the moment when all the VMs of the system are again in active state. The aim of these experiments is to evaluate whether the proposed framework could be successfully adopted to increase the performances of the X1.V1 platform on the Cloud, e.g., by addressing the issues described by the two reference examples presented in Section 3.

In general, we can say that the porting of the X1.V1 platform on the Cloud through the proposed framework is beneficial when the time required to perform the switch between two configurations is low enough to be compensated by the better system performance due to the new configuration. In particular, to address the issue described in *Example 1*, the time required to switch to a configuration which allocates more resources to the VM(s) that performs the statistics should be compensated by a faster execution of the statistics procedure. In other words, the time T_N required to carry out the statistic task exploiting the new configuration should be lower than the sum of time T_I required to carry out the same task exploiting the initial configuration and twice the time T_S required to perform the configuration switch ($T_N + 2T_S < T_I$). We consider twice the time required to perform the configuration switch ($2T_S$) because we take into account also the time to restore the initial configuration, since the new configuration could be inadequate for the daily operation of the system, supposing that the two configuration switches take the same time. To address the issue described in *Example 2*, instead, the time required to switch to a configuration which allocates more resource to the VM(s) running the Prescription service should be reasonably low in order to give a prompt response to users (GPs, Pharmacists, Specialists and, overall, the patients) that are waiting for service completion.

The physical testbed where we performed the first set of experiments consists of one machine equipped with a Intel i7 2600 3.4 Ghz CPU with 8 cores and 8GB of RAM memory which hosts the Controller Node, and one machine equipped with an AMD Opteron 6380 2.5 Ghz CPU with 16 cores and 32 GB of RAM memory, which

[12] http://collectd.org/

hosts the compute node. The operating system installed on these machines is Linux Ubuntu 12.04 LTS (Precise Pangolin), with 3.11.0-26-generic 64bit kernel. The OpenStack version used is Havana with the components Keystone (Security manager), Glance (Image manager) and Nova (VM manager). The Controller node hosts the following modules: keystone, glance-registry, glance-api, nova-api, nova-cert, nova-consoleauth, nova-scheduler, nova-conductor, nova-novncproxy. The Compute nodes host nova-compute and nova-network.

The allocation of the resources to the VMs is defined by the following three VM flavors:

- *Small* (1 VCPU, 1GB RAM)
- *Medium* (1 VCPU, 1,5GB RAM)
- *Large* (1 VCPU, 2GB RAM)

Although our architecture states that the configuration files follow the OVF standard, we found out that the current release of OpenStack does not provide a stable support for OVF yet. Hence, the current implementation adopts a custom format to represent the configuration files, which will be replaced by a customization of the OVF format as soon as possible. The custom format used for the experiments has a XML structure with the essential information for each VM.

To perform our tests we defined four distinct configurations: *Normal, Interactive, Night*, and *Power* which differ one from another for the flavors choosen for each VM of the system. In fact, in the *Normal* configuration the flavor of all the VMs is *Medium*, while in the *Power* configuration, all the VMs are *Large*. Moreover, supposing to have *N* VMs, in the *Interactive* (respectively, *Night*) configuration, the flavor of the first *N/2* VMs is *Large* (respectively, *Small*) and the flavor of the remaining VMs is *Small* (respectively, *Large*). Table 6.1 reports these configurations in case of 4 VMs.

Table 6.1
Configurations in case of 4 VMs

	Normal	Interactive	Night	Power
VM1	Medium	Large	Small	Large
VM2	Medium	Large	Small	Large
VM3	Medium	Small	Large	Large
VM4	Medium	Small	Large	Large

The full amount of resources required by the *Normal, Interactive*, and *Night* configurations is the same (4 VCPUs and 6GB of RAM memory in the example of Table 6.1), while the *Power* configuration requires more resources (4 VCPUs and 8GB of RAM memory in the same example). Figures 6.1 and 6.2 show the time required to switch from one of the four configurations to another for each couple of them, varying the total number of VMs of the system. The VM image we used for this set of experiments is the Ubuntu Cloud image, which is "a pre-installed disk image that has been customized by Ubuntu engineering to run on cloud-platforms such as OpenStack" equipped with an Ubuntu Server 12.04 LTS (Precise Pangolin)[13]. The Ubuntu Cloud image size is 248MB.

The results in Figures 6.1 and 6.2 clearly show that the time required to switch from one configuration to another depends on the number of VMs whose flavor is changed. In particular, Figure 6.1 shows the time required to switch from a configuration to another in those cases where this switch requires to change the flavor of half of the VMs of the system. As an example, in the case of 8 VMs the switch from the *Interactive* to the *Power* configuration requires to change the flavor of 4 VMs from *Small* to *Large*. Figure 6.2, instead, shows the time required to switch from a configuration to another in those cases where this switch requires to change the flavor of all the VMs of the system. As an example, in the case of 8 VMs the switch from the *Normal* to the *Power* configuration requires to change the flavor of 8 VMs from *Medium* to *Large*.

For any number of VMs, we notice that the time to perform the configuration switchs requiring to change the flavor of half of the VMs (Figure 6.1) is considerably less than the time to perform the configuration switches that require to change the flavor of all the VMs (Figure 6.2). For example, in the case of 8 VMs, the time to switch from the *Interactive* configuration to the *Power* one is about 20 seconds, while the time to switch

[13] https://cloud-images.ubuntu.com/precise

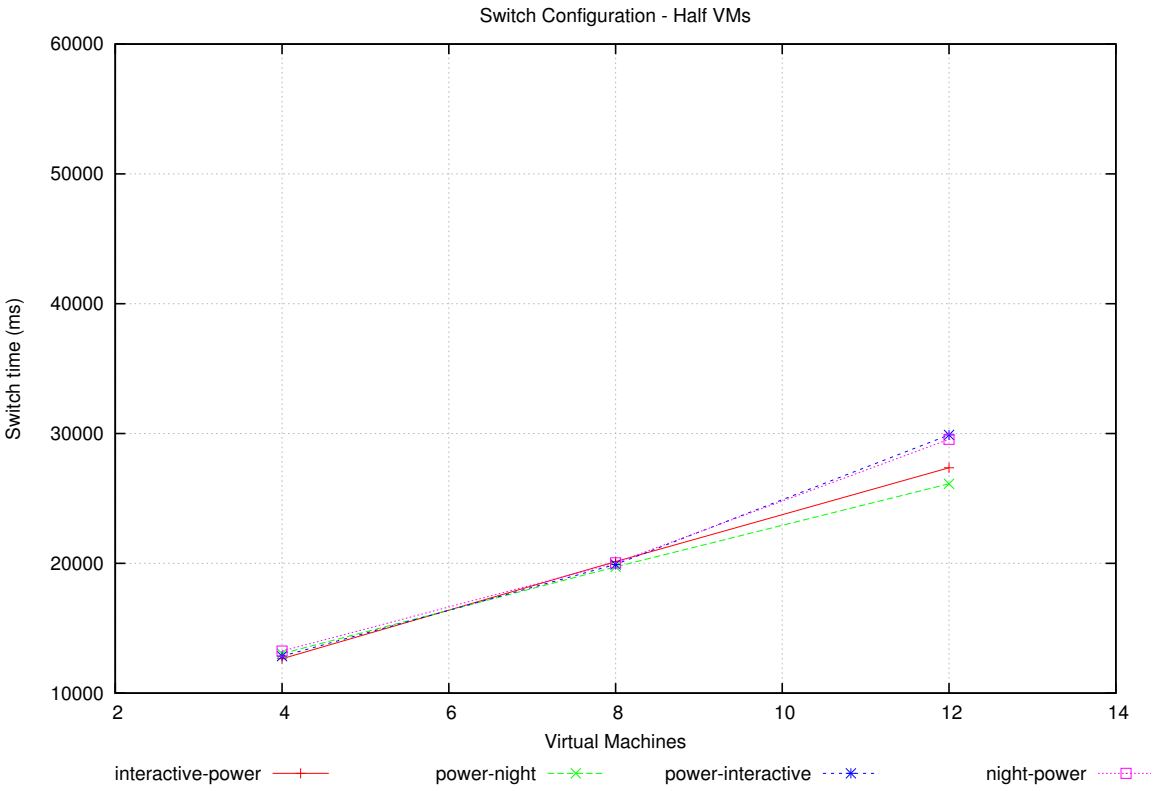

FIG. 6.1. *Time to perform a switch involving half of the VMs (Ubuntu Cloud image)*

from the *Normal* to *Power* one is about 31 seconds. Moreover, we also notice that increasing the number of VMs, the time required to perform any kind of switch increases. In particular, we notice that, doubling the number of VMs whose flavor is changed (either doubling the number of the VMs of the system or performing a configuration switch that require to change the flavor of all the VMs instead of half of them), the time required to perform the whole switch procedure is considerably higher, although the switch procedure does not take twice the time. For example, the time to perform the switch from the *Normal* configuration to the *Power* one in case of 4 VMs is about 19 seconds, and in case of 8 VMs is about 31 seconds. We estimated that, on average, the time required to change the flavor of N VMs is 156% the time required to change the flavor of $N/2$ VMs.

Furthermore, we notice that the maximum switch time in case of 4 VMs is about 20 seconds, while in case 12 VMs is about 47 seconds. This means that switching from a configuration that is not suitable to perform a given task to one that is more suitable for this task is convenient when the second configuration allows to save at least 20 seconds in case of 4 VMs and 47 seconds in case of 12 VMs. However, if the second configuration is not suitable for the normal operation of the system, we should consider twice the switch time i.e., respectively, 40 and 94 seconds, because we need to restore the initial configuration.

The results in Figure 6.3 show the time to perform a configuration switch exploiting another VM image, i.e., Ubuntu 12.04 Desktop image, whose size is 1.4GB. The figure shows the time required to perform both the configuration switches that involve half the VMs of the system and all the VMs of the system. We notice that the results in Figure 6.3 confirm the comments we made for Figures 6.1 and 6.2, and that, in the worst case, the time required for a configuration switch is about 20 seconds on 4 VMs and less than 50 seconds on 12 VMs. Since the previous set of experiments has been executed exploiting one compute node only, no VM migration occurred.

Hence, we performed another set of experiments on a different testbed, which consists of four machines. This testbed includes one machine equipped with a Intel i7 2600 3.4 Ghz CPU with 8 cores and 8GB of RAM

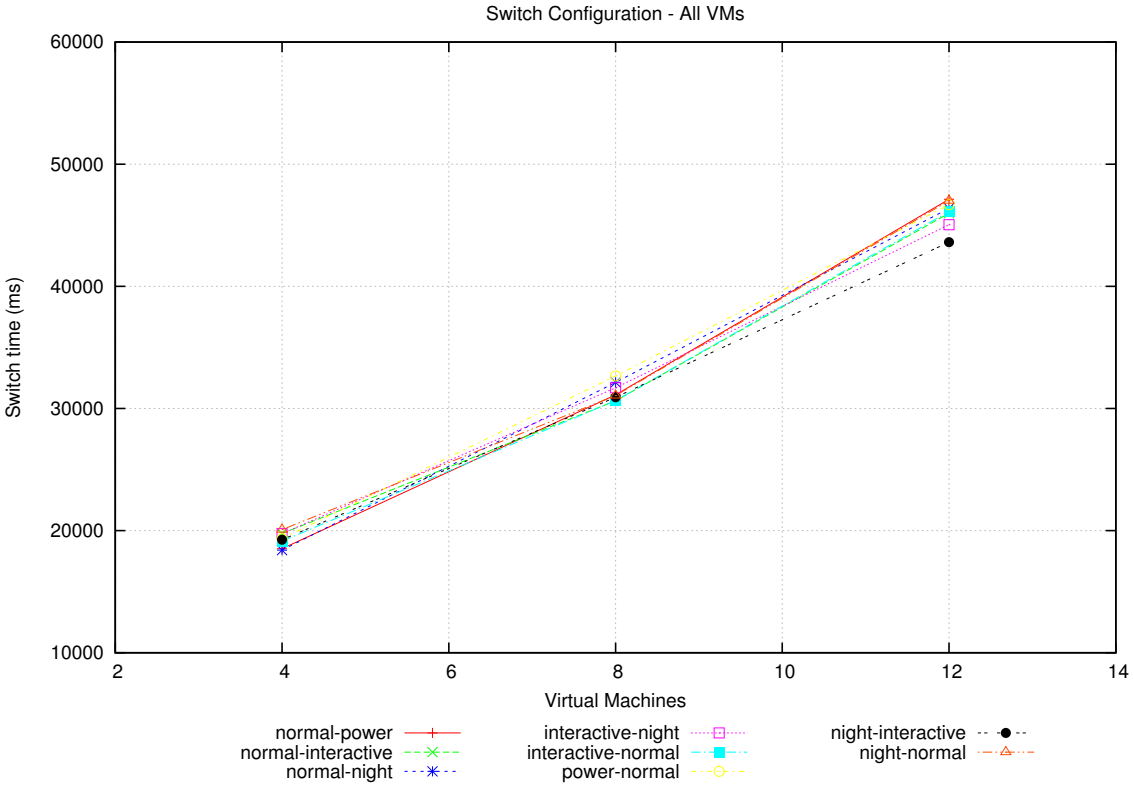

FIG. 6.2. *Time to perform a switch involving all of the VMs (Ubuntu Cloud image)*

memory which hosts the Controller Node, and three machines which are exploited as compute nodes: the first machine (A) is equipped with an AMD Opteron 6380 2.5 Ghz CPU with 16 cores and 32 GB of RAM memory, the second machine (B) is equipped with an Intel i5 750 2,66 Ghz CPU with 4 core and 8 GB of RAM memory, and the third machine (C) is equipped with Intel i5 760 2,8 Ghz CPU with 4 core and 8 GB of RAM memory. The operating system installed on all these machines is Linux Ubuntu 12.04 LTS (Precise Pangolin), with 3.11.0-26-generic 64bit kernel. These experiments also consider a different allocation of the resources for the three VM flavors:

- *Small* (1 VCPU, 1GB RAM)
- *Medium* (2 VCPU, 1,5GB RAM)
- *Large* (4 VCPU, 3GB RAM)

Moreover, the VM image we exploited for all the VMs is Ubuntu Server 12.04 LTS (Precise Pangolin), and the image size is 3GB.

Figure 6.4 shows the time required to perform the configuration switch from the *Interactive* to the *Power* configuration, which changes the flavor of half the VMs of the system, and the time required to switch from the *Normal* configuration to the *Power* one, which involves all the VMs of the system. When the system consists of 4 VMs, the time to switch from the *Interactive* to the *Power* configuration is about 17 seconds, while the time to switch from the *Normal* to the *Power* configuration is about 28 seconds and the VMs, that before the switch were running on machine A, did not migrate on other machines. The results for 4 VMS are slightly greater with respect to the ones obtained from the previous set of experiments. We think that this could be due to the fact that the flavour used in the second set of experiments exploits a larger set of resources (VCPUs and RAM memory).

Instead, when the system consists of 8 VMs, the switch from the *Interactive* to the *Power* configuration, on average, takes about 156 seconds. This time is considerably larger with respect to the one in Figure 6.3,

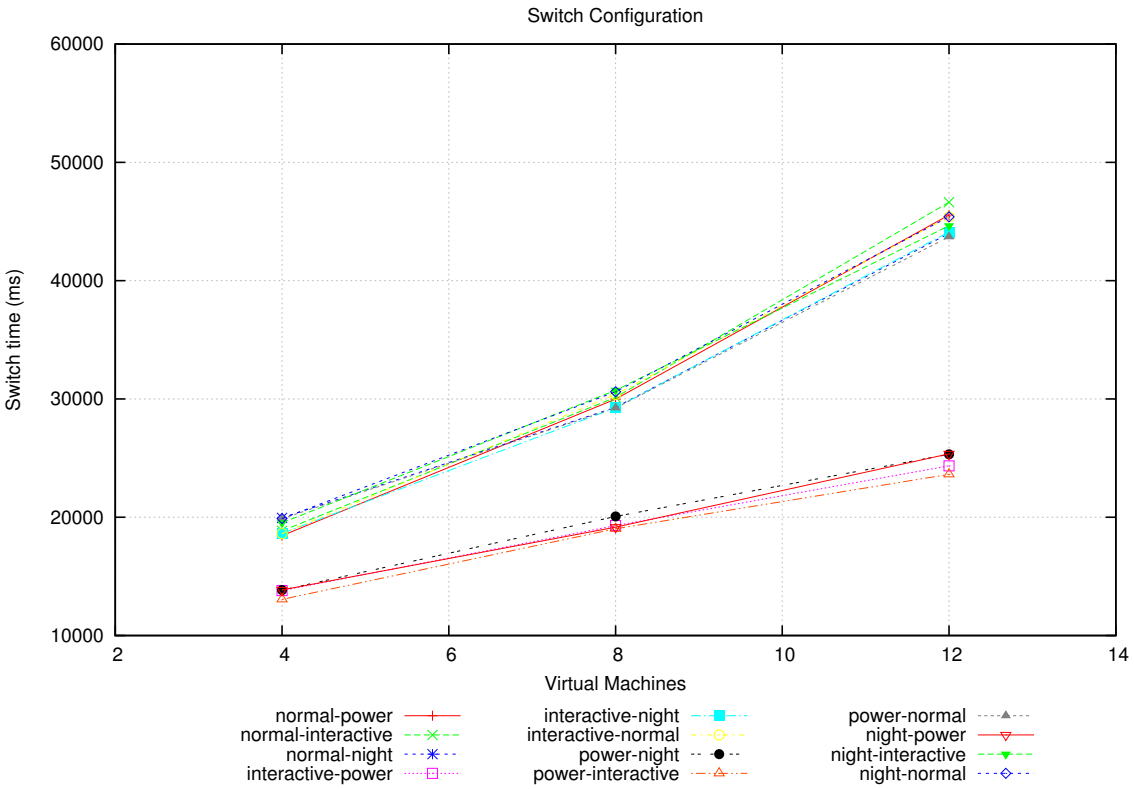

FIG. 6.3. *Time to perform a switch (Ubuntu Desktop image)*

and the main reason is that, on average, about 2 VMs (2,15) migrated from one compute node to another. In particular, taking into account one specific experiment, before the switch 5 VMs were running on machine A, 2 VMs were running on machine B, and 1 VM was running on machine C. The switch caused the migration of 3 VMs, and after the switch all the VMs were running on machine A. The time required for this specific experiment was 222 seconds. In another experiment, 6 VMs were allocated on machine A and 2 VMs were allocated on machine B before the switch. The switch involved the migration of 2 VMs, and took about 137 seconds. After the switch all the VMs were allocated on machine A. In a further experiment, before the switch all the VMs were running on machine A, and after the switch all the VMs were still running on machine A. In this case, the time required to perform the switch was about 32 seconds. As far as concerns the switch from the *Normal* to the *Power* configuration, the average execution time is almost the same, i.e., about 158 seconds, and the average number of migrated VMs is about 2 (2,21). Finally, the results of the experiments performed with 12 VMs show that the *Interactive* to *Power* configuration switch requires, on average, 358 seconds and involves the migration of 5 VMs, while changing the configuration from *Normal* to *Power* requires 378 seconds and involves the migration of 6 VMs.

These results show that, in case of migration, the time required to perform the configuration switch mainly depends on the number of VMs that are migrated, and the number of VMs whose flavor is changed is not important in this case. In turn, the number of VMs that are migrated from one physical machine to another depends on the original allocation of the VMs to the physical machines, and on the allocation policy of OpenStack. This explains, for instance, why the time required for the configuration switch in case of 8 VMs is almost the same in the case where the flavor of 4 VMS is changed and in the case where the flavor of 8 VMs is changed, i.e., because the average number of migrated VMs is almost the same. Moreover, we performed a further set of experiments on 4 VMs exploiting the same VM image described previously and the following flavors:

- *Small* (1 VCPU, 1GB RAM)

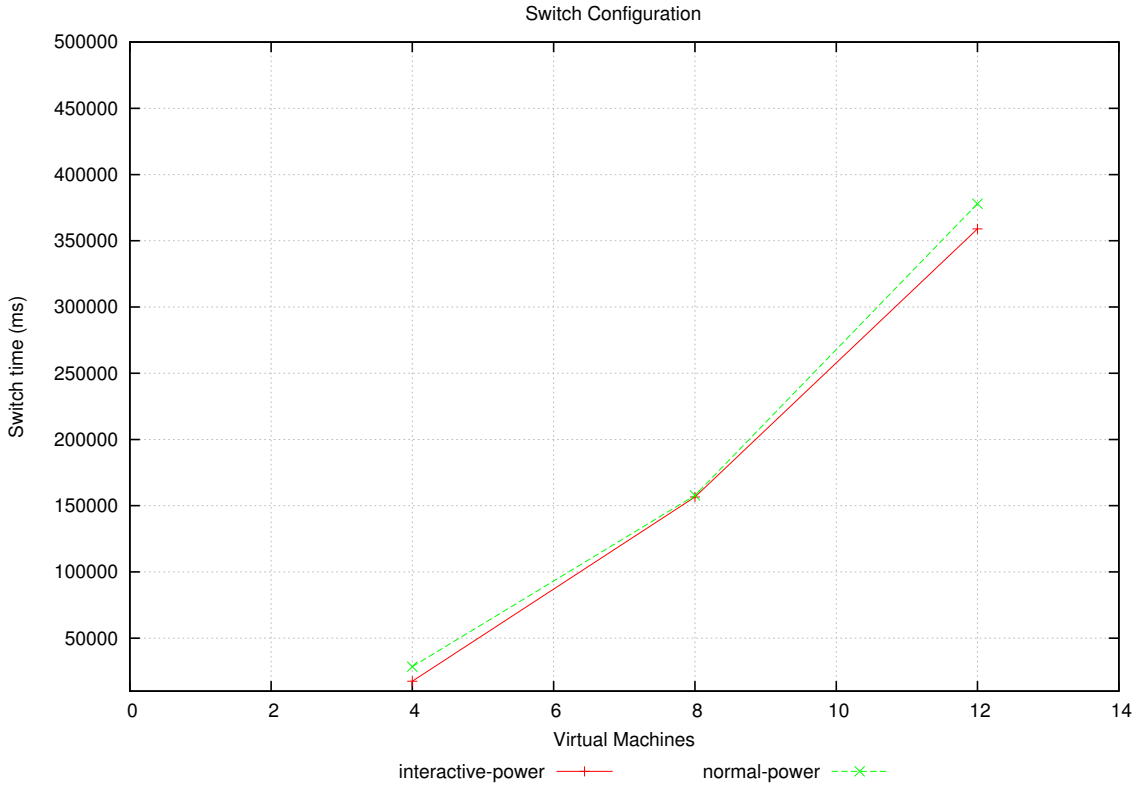

FIG. 6.4. *Time to perform a switch (Ubuntu Server image)*

- *Medium* (4 VCPU, 4GB RAM)
- *Large* (6 VCPU, 6GB RAM)

The average switching time was 148 seconds and, on average, the number of migrated VMs was 2 (1,9). This result confirms that the switching time mainly depends on the number of VMs that are migrated. As a matter of fact, the switching time of this experiment is almost the same as the one obtained in the previous set of experiments with 8 VMs (156 seconds) because the number of VMs that migrated is almost the same (about 2) in the two experiments.

7. Conclusion. This paper proposed a tool, the X1.V1 Cloud Manager, which optimizes the resource utilization for the execution of the X1.V1 platform on the Cloud by easing the exploitation of the Cloud elasticity. In fact, the X1.V1 Cloud Manager allows to change the allocation of the Cloud resources to the VMs according to some predefined configurations, in order to reduce the response time of the X1.V1 services in case of computational peaks, to reduce the completion time of some heavy computational tasks, and to save resources when the overall load of the platform is low. Since the resource requirements of the X1.V1 platform is known, the configuration switch to be performed during the daily operations can be planned in advance. However, some configuration switches can be manually initiated by the administrators to resolve unexpected computational peaks. Moreover, as a future work, we plan to integrate a further component in our framework in order to deal with unpredictable variation of the computational load of the VMs, i.e., to automatically detect, or even predict, when a new configuration should be adopted to achieve better performance.

The X1.V1 Cloud Manager security support includes proper authentication and authorization mechanisms, in order to avoid that unauthorized subjects change the Cloud resource allocation, since this could reduce the performance of the X1.V1 platform, increase the cost of the computation, and could even result in a Denial of Service. The paper presented two sets of experimental results which prove that, when no VM is migrated

from one compute node to another, the time required to perform a configuration switch is quite low, even in case of a large number of VMs. When some VMs are migrated from one compute node to another, instead, the configuration switch time increases considerably. Since the VMs migration affects the configuration switch time, which depends on the size of the VMs that are migrated, the X1.V1 Cloud manager will be configured in a way such that only the VMs with small images are migrated. Hence, the VMs storing the EHR data are not be migrated, while the VMs implementing some of the services provided by the X1.V1 platform could be migrated.

As ongoing work, we are deploying our framework in the main data center of the Dedalus SpA company (which is located in Avellino, Italy) in order to perform some simulations in a real scenario, trying also to evaluate the benefits resulting from changing configuration in some specific load distribution.

As future work, we plan to study the adoption of an advanced authorization model, the Usage Control Model [24] to regulate the usage of the X1.V1 VMs, as described in [34], [17]. As an example, the usage control authorization system could restore the previous (or a standard) configuration when the administrator that updated the resource allocation with a new configuration loses the right required to perform this switch. This policy could be adopted when the new configuration involves additional costs for the user.

Acknowledgment. This work was partially supported by the Italian project: "Piattaforme Cloud Sicure per applicazioni critiche: il caso del Fascicolo Sanitario Elettronico (PICs): progetto POR CRO FSE 2007-2013 Asse IV Capitale Umano" funded by Regione Toscana.

REFERENCES

[1] A. Ali-Eldin, J. Tordsson, and E. Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 204–212. IEEE, 2012.

[2] A. Bahga and V.K. Madisetti. A cloud-based approach for interoperable electronic health records (EHRs). *IEEE Journal of Biomedical and Health Informatics,*, 17(5):894–906, 2013.

[3] L. Beernaert, M. Matos, R. Vilaça, and R. Oliveira. Automatic elasticity in openstack. In *Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management (SDMCMM 12)*, pages 2:1–2:6. ACM, 2012.

[4] S. Di Bona. Benefits of cloud compunting in EHR implementation. In *Proceedings of the Global Forum 2012 : SHAPING A CONNECTED DIGITAL FUTURE, Stockholm (Sweden), November 12th-13th, 2012*. Dedalus SpA, 2012.

[5] G. Brataas, E. Stav, S. Lehrig, S. Becker, G. Kopčak, and D. Huljenic. Cloudscale: Scalability management for cloud systems. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, ICPE '13, pages 335–338, New York, NY, USA, 2013. ACM.

[6] Y. Chen, J. Lu, and J. Jan. A secure EHR system based on hybrid clouds. *Journal of Medical Systems*, 36(5):3375–3384, 2012.

[7] Cloud Security Alliance. Security guidance for critical areas of focus in cloud computing v3.0, 2011.

[8] Cloud Security Alliance. The notorious nine. cloud computing top threats in 2013, February 2013.

[9] DMTF. Open Virtualization Format Specification. Version: 2.1.0. Distributed Management Task Force, 2013.

[10] ENISA. Cloud computing - benefits, risk and recommendations for information security, 2009.

[11] F. Frisch. The identity & access management (r)evolution. Federation and attribute based access control. Axiomatics AB, 2014.

[12] G. Galante and L.C.E. de Bona. A survey on cloud computing elasticity. In *Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on*, pages 263–270, 2012.

[13] Z. Gong, X. Gu, and J. Wilkes. Press: Predictive elastic resource scaling for cloud systems. In *Proceedings of the 2010 International Conference on Network and Service Management (CNSM 10)*, pages 9–16, 2010.

[14] N.R. Herbst, S. Kounev, and R. Reussner. Elasticity in cloud computing: What it is, and what it is not. In *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*, pages 23–27. USENIX, 2013.

[15] C.N. Hoefer and G. Karagiannis. Taxonomy of cloud computing services. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1345–1350, Dec 2010.

[16] M. Kuo, Y. Guo, and T. Sahama. Cloud computing for healthcare research information sharing. In *Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom 12)*, pages 889–894, 2012.

[17] A. Lazouski, G. Mancini, F. Martinelli, and P. Mori. Usage control in cloud systems. In *Proceedings of the 7th International Conference for Internet Technology and Secured Transactions (ICITST12)*, pages 202–207, 2012.

[18] S. La Manna. The vision of dedalus for interoperability innovation in the ehealth sector. In *Proceedings of the Global Forum 2013 : DRIVING THE DIGITAL FUTURE, Opportunities for Citizens and Businesses, Trieste (Italy), October 28th-29th, 2013*. Dedalus SpA, 2013.

[19] M. Matos, Jr. A. Correia, J. Pereira, and R. Oliveira. Serpentine: Adaptive middleware for complex heterogeneous distributed systems. In *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC 08)*, pages 2219–2223. ACM, 2008.

[20] P. Mell and T. Grance. The NIST definition of cloud computing. recommendation of the national institute of standards and technology, 2011.
[21] OASIS. SAML 2.0 profile of XACML version 2.0, August 2010.
[22] OASIS. XACML v3.0 core and hierarchical role based access control (RBAC) profile version 1.0, March 2010.
[23] OASIS. eXtensible Access Control Markup Language (XACML) version 3.0, January 2013.
[24] J. Park and R. Sandhu. The $UCON_{ABC}$ usage control model. *ACM Transactions on Information and System Security*, 7:128–174, 2004.
[25] B.E. Reddy, T.V.S. Kumar, and G. Ramu. An efficient cloud framework for health care monitoring system. In *Proceedings of the International Symposium on Cloud and Services Computing (ISCOS 2012)*, pages 113–117, 2012.
[26] N. Roy, A. Dubey, and A. Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. *2013 IEEE Sixth International Conference on Cloud Computing*, 0:500–507, 2011.
[27] R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.
[28] J. Sermersheim. Lightweight directory access protocol (LDAP): The protocol. IETF Request for Comments: 4511, 2006.
[29] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes. Cloudscale: Elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC 11)*, pages 5:1–5:14. ACM, 2011.
[30] L.M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, December 2008.
[31] G. von Laszewski, J. Diaz, W. Fugang, and G.C. Fox. Comparison of multiple cloud frameworks. In *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD 2012)*, pages 734–741, 2012.
[32] M. Wahl, H. Alvestrand, J. Hodges, and R. Morgan. Authentication methods for LDAP. IETF Request for Comments: 2829, 2000.
[33] X. Wen, G. Gu, Q. Li, Y. Gao, and X. Zhang. Comparison of open-source cloud management platforms: Openstack and opennebula. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 2457–2461, 2012.
[34] X. Zhang, M. Nakae, M.J. Covington, and R. Sandhu. Toward a usage-based security framework for collaborative computing systems. *ACM Transactions on Information and System Security*, 11(1):3:1–3:36, 2008.

Edited by: Jesus Carretero
Received: August 31, 2014
Accepted: January 21, 2015