Platform to Ease the Deployment and Improve the Availability of TRENCADIS Infrastructure

Damià Segrelles[1], Miguel Caballer[1], Erik Torres[1], Germán Moltó[1], Ignacio Blanquer[1]

Instituto de Instrumentación para Imagen Molecular (I3M), Universitat Politècnica de València, València, Spain
dquilis@dsic.upv.es, micafer1@upv.es, ertorser@upv.es, gmolto@dsic.upv.es, iblanque@dsic.upv.es

Abstract. TRENCADIS is a Grid infrastructure to store and to process large amounts of medical images and its associated data in DICOM objects. This system enables radiologists to effectively group, search and manipulate images and structured reports in order to relate clinical findings and to be of practical value in the diagnosis and treatment of diseases. The paper presents a new platform for the deployment of TRENCADIS infrastructures, using virtualization and Cloud computing techniques. The presented platform avoids intrusive deployment of services to reduce the amount of effort required to install and maintain new TRENCADIS services. Also, it provides mechanisms to monitor and handle performance and reliability requirements by elastically provisioning computational resources from the Cloud to cope with increased demand of the platform.

1 Introduction

Modern medicine cannot be conceived without medical imaging. These techniques play a major role in the diagnosis and treatment of diseases and they are gaining in importance in the prevention and control of epidemics. Hospitals have made large inversions to implement Picture Archiving and Communication Systems (PACSs) and Radiology Information Systems (RISs). These technologies provide storages of medical images, but they are often limited in their access to Distributed Computing Infrastructures (DCIs), such as Grid and Cloud computing environments, which are reducing costs of computing service provision [1], fostering innovative practices and accelerating their adoption by the healthcare professionals [2]. Reaching the level of security that is necessary to ensure the protection from disclosure of the identity of patients is a difficult task. Policy decision-makers have recently become aware of the value of DCIs to improve public health and they are lowering the barriers to Cloud adoption in the European Community [3].

In the last years, many authors have anticipated this trend by developing prototypes that make use of different DCIs to store and process medical images, e.g. [4]. TRENCADIS [5] is an example of the use of Grid computing infrastructures to store and to process large amounts of medical images, in a secure way. This system enables radiologists to effectively group, search and manipulate Digital Imaging

and Communications in Medicine [6] (DICOM) images and structured reports in order to relate clinical findings and to be of practical value in the diagnosis and treatment of diseases. DICOM is the accepted standard format for medical image storage and transfer. Since the introduction of TRENCADIS, several hospitals have reported the value within their organizational context of this technology as a tool for improving the access to DICOM objects [5], and [7]. However, in all these cases the deployment and maintenance of TRENCADIS required a considerable investment of time and effort, mainly because of the intrinsic complexity of Grid systems and also due to the network security restrictions imposed on the hospitals. This fact makes difficult to extend or adapt a particular deployment to meet new challenges, such as an unexpected increase in demand or in the number of images stored in the system. This paper was motivated by these previous results, which provided the basis for new TRENCADIS deployment strategies.

The objective of this paper is to present a new platform for the deployment of TRENCADIS-based applications, using virtualization and Cloud computing techniques. The presented platform avoids intrusive deployment of services to reduce the amount of effort required to install and maintain new TRENCADIS sites. Also, it provides mechanisms to monitor and handle performance and reliability requirements together, allocating and de-allocating different computational resources from the Cloud, such as virtual machines, as required by the applications. Finally, the platform was designed to be portable to other application domains that also rely on Grid computing.

In the case of reliability, the approach in this paper is to use proactive redundancy in the instances of the services that supports critical functions, so that a certain number of faults can be tolerated. In this way, the services are replicated to multiple resources in the Cloud and their management and monitoring is automated to speed-up the creation of new replicas to replace failed ones, so that reliability is maintained.

The rest of this paper is structured as follows. Section 2 analyzes the current tools which allows to automate the deployment of Grid applications to Cloud computing environments. Section 3 presents a comprehensive analysis of the software components and customized required configurations of all TRENCADIS services, from the point of view of their security, availability and performance requirements. Section 4 describes the platform that was developed to automate the deployment of these services on Cloud computing environments, describing the conditions under which the presented platform can be used as a basis for application deployment. Section 5 presents a test deployment to validate the presented platform and to illustrate its capabilities and benefits respect to a traditional TRENCADIS deployment. Finally, section 6 presents the conclusions of the paper, as well as the future lines of work.

2 State of the Art

There can be found works in the literature that aim at easing the process of software deployment on different platforms. For example, Puppet [9] and Chef [10] are open-source configuration management systems that are typically used to

deploy applications on provisioned computational resources. This systems can be used to create and manage configurations rules (recipes) that describe a series of resources, such as software libraries that should be installed, services that should be running or files that should be written, on a particular (virtual) machine, thus automating the application deployment process.

With the success of Cloud computing environments like Amazon EC2 [11], many open-source tools have been developed to deploy popular systems to these infrastructures. For example, Apache Whirr [12] is a tool for running Apache Hadoop and related services, such as Cassandra or HBase, in Amazon EC2. In general, although these tools can be adapted to other Cloud providers or other systems (with more or less effort), the use of generic tools seems to be more convenient for our purposes.

In the case of applications that depend on a programming model, such as message-passing, MapReduce or Grid programming models, different approaches are required to deploy the applications. This is a very common requirement, especially in scientific applications. To this end, several platforms support not only the configuration management, but also the specific programming models and languages. CloudFoundry [13] is an example of open-source application deployment system that supports several popular application development frameworks, such as the Spring Framework, Ruby on Rails or Grails. It also provides a set of services to the application developers that includes relational (e.g. MySQL or PostgreSQL) and NoSQL (e.g. Redis) database services, or messaging services (e.g. RabbitMQ). AppScale [14] is an open-source implementation of the Google App Engine [15] Cloud computing technology. It provides Neptune, an extension of the Ruby programming language that supports Message Passing Interface (MPI) and MapReduce programming models. AppScale can be used to automate the configuration and deployment of existing HPC applications to Cloud.

In contrast to "pure" configuration management systems, such as Puppet or Chef, Cloud platforms, such as CloudFoundry or AppScale, have the advantage of integrating support for specific programming models and development frameworks. Often, this support for development that is present in Cloud frameworks limits the flexibility with which the virtual machines can be configured to meet application requirements. For the purposes of this work, this is a serious limitation that makes difficult to configure the services with the desired management and security properties.

On the other hand, to our knowledge, there is no mention on the literature of an efficient and affordable tool which allows to automate the deployment of Grid applications to Cloud computing environments, and their configuration to be used in a secure way, with constraints to minimize the overall system downtime.

3 Analysis of TRENCADIS Infrastructure Services

Figure 1 shows the TRENCADIS infrastructure deployment model. This infrastructure is composed by a set of services based on Grid technologies which are integrated in a Virtual Organization (VO). There are two categories, the CORE

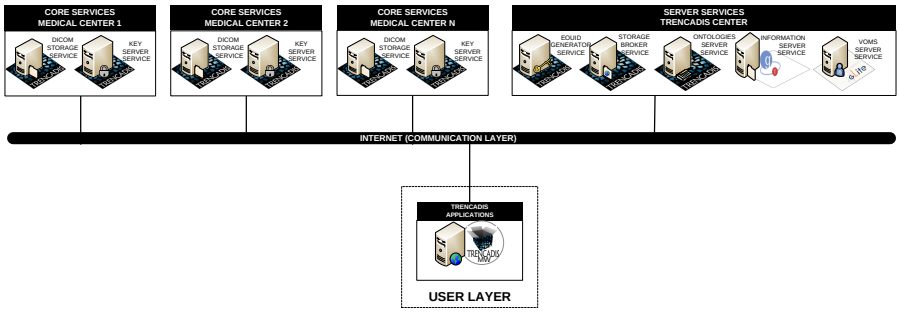

Fig. 1. Deployment model of TRENCADIS infraestructure.

services and SERVER services. Scientific Linux 6.7 OS is recommended for all of them, although another compatible GNU/Linux OS could be used.

3.1 CORE Services

The CORE services are the DICOM Storage services and the Key Server services. At least one instance of each type should be deployed in each medical center involved in the VO. The **DICOM Storage Service** is composed of the following four software components. The implementation depends on the underlying technology that is used to support the functionality of the component:

- *Base Toolkits*. This component is the base of all services in TRENCADIS that rely on Grid services. It is formed by the Globus 4 toolkit, JDK 1.6, and Ant 1.8. It allows implementing and deploying Grid services. This component does not require a customized configuration.
- *DICOM Storage Grid Service*. This component is deployed on the service container provided by Globus 4. This Grid service uses different versions of APIs to connect to the Indexer and Backend components, depending on how these are implemented. Depending on the version of the APIs used, a customized configuration of each API employed is required. Moreover, it is needed to configure the Grid service for integration in the VO.
- *Indexer*. This component enables to index data contained in DICOM structured reports and can be supported by SQL relational databases (PostgreSQL) or by the Grid component gLite AMGA Server [16]. Currently, support for NoSQL databases, such as Neo4j [18], is being implemented, although it is not operational yet. This component does not require a customized configuration because it is handled directly through an API by the component DICOM Storage Grid service.
- *BackEnd*. This component stores encrypted DICOM images and associated DICOM structured reports. The backend can be supported by a GridFTP server, Grid components available on gLite (LFC and SE) [17], a File System or a SQL relational database (PostgreSQL). Currently, support for integrating CDMI interfaces [19] is being implemented to use Cloud backends, but it is not

operational yet. This component does not require a customized configuration because it is handled directly through an API by the component DICOM Storage Grid service.

The **Key Server Service** is composed of the following software components:

- *Key Server Grid Service*. This component is deployed on the service container provided by Globus 4. This Grid service uses an API for connecting to the SQL Key Database, installed in the backend. Moreover, it is needed to configure the Grid service for integration in the VO.
- *BackEnd*. This component stores the keys used for encrypting/decrypting DICOM data. This is the same component presented above, but in this case it can only be supported by a SQL relational database (PostgreSQL).
- *SQL Keys Database*. This component is a relational database installed into the backend (PostgreSQL). The database has a predefined structure and needs to be created and configured when the backend is deployed.

3.2 SERVER Services

Server services are a set of the five services, which have to be deployed in one center (TRENCADIS Center). This center is external to medical centers involved in the VO. The **EOUID Generator Service** is composed of one software component. This component is deployed on the service container provided by Globus 4 and implements the logical for generating the Encrypted Object Unique Identifier (EOUID). Moreover, it needs to be configured for integrating in the VO. The database has a predefined structure and needs to be created and configured when the backend is deployed. The **Storage Broker Service** is composed of one software component. This component is deployed on the service container provided by Globus 4 and implements the logic for distributing queries among the DICOM Storage services involved and retrieving the results. Moreover, it needs to be configured for integration in the VO. The **Ontologies Server Service** is composed of these software component:

- *Ontologies Server Grid Service*. This component is deployed on the service container provided by Globus 4. This Grid service uses an API for connecting with the SQL Ontologies Database, installed in the backend. Moreover, it is needed to configure the Grid service for integration in the VO.
- *BackEnd*. This component store the Ontologies used for organizing the DICOM data. This is the same component presented in the Key Server service.
- *SQL Ontologies Database*. This component is a relational database installed into the backend (PostgreSQL). The database has a predefined structure and needs to be created and configured when the backend is deployed.

The **Information Server Service (ISS)** is part of the Monitoring and Discovery System (MDS4) of Globus 4. Therefore, it is only composed by the component base toolkits, which have been presented above. Moreover, it requires to configure the service for integration in the VO. The **VOMS Service** is part of the gLite Middleware for Grid Computing [20]. Also it is needed to configure the service for integration in the VO.

4 TRENCADIS Cloud Deployment Platform

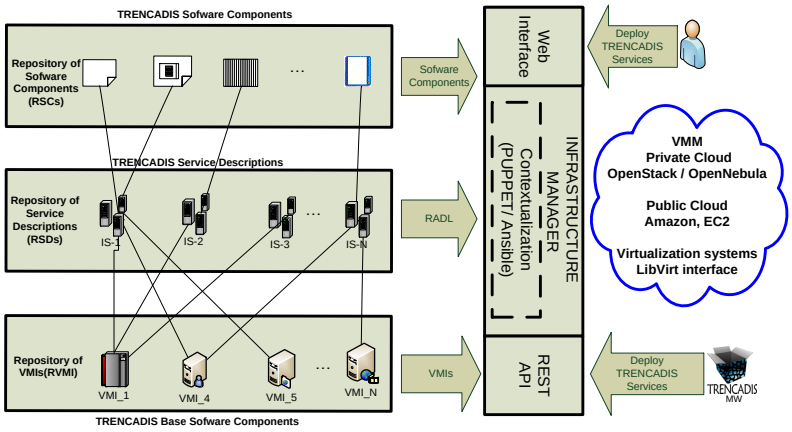

Fig. 2. TRENCADIS Cloud Deployment Platform.

The components of the platform architecture are described in Figure 2 and described in the following subsections:

4.1 Repository of VMIs (RVMI)

Table 1. Created VMI for TRENCADIS Infrastructure

ID	VMI Name	VMRC Metadata Description
1	Base Toollkits	Scientific Linux 5.7; Globus Toolkit 4.2.1; JDK 1.6.0.35; Ant 1.8.2
2	VOMS Service	Scientific Linux 5.7; VOMS Service
3	PostgreSQL	Scientific Linux 5.7; Postgres 8.4.9
4	LFC	Scientific Linux 5.7; gLite LFC
5	SE	Scientific Linux 5.7; gLite SE
6	File System	Scientific Linux 5.7
7	GridFTP	Scientific Linux 5.7; GridFTP

RVMI is implemented with the Virtual Machine image Repository and Catalog (VMRC) [21]. VMRC is a software component that enables users to index and store virtual Machine Images (VMIs) together with metadata descriptions about the capabilities of each VMI in terms of CPU architecture, hypervisor for which it was built, OS, applications, etc... This enables users to share and reuse VMIs for

different TRENCADIS services. A client-side API is provided in order to query for the most appropriate VMIs that satisfy a given set of both hard and soft requirements. Whereas the hard ones must be satisfied by VMIs, the soft ones allow obtaining a ranked list of VMIs depending on the degree of satisfaction of the requisites. For example, one could ask for a VMI created for the KVM hypervisor that has Scientific Linux greater than 5.7, the Globus Toolkit 4 and Java Development Kit (hard requirements) but it would be desirable to have Java 1.6.35 toolkit (soft requirement). Therefore, the VMRC enables to catalog a set of base VMIs from which other VMIs can specifically customized in order to fit a particular deployment.

Table 1 lists the created VMIs for deploying TRENCADIS infrastructures and their VMRC metadata descriptions. These VMIs contain the base software components that can be reused to deploy the different TRENCADIS services.

4.2 Repository of Sofware Components (RSCs)

This repository is a directory with installations files of software components that are needed for creating and deploying the TRENCADIS services.

These software components have to be installed on top of the VMIs listed in table 1. To provide reliability and auto scaling of services a specific configuration is needed that depend on the Cloud used. Table 2 shows the files and their type. For example, Grid Archive Files (Gar) are needed to deploy Grid services in the Globus Toolkit 4 container.

Table 2. Installation Files for TRENCADIS Infrastructure

ID	Instalation File	Type of File
1	DICOM Storage Grid Service	Gar
2	Key Server Grid Service	Gar
3	EOUID Generator Grid Service	Gar
4	Storage Broker Grid Service	Gar
5	Ontologies Server Grid Service	Gar
6	TRENCADIS Java API Indexer	Jar
7	TRENCADIS Java API Data Backend	Jar
8	TRENCADIS Java API SQL Keys DB	Jar
9	TRENCADIS Java API SQL Ontologies DB	Jar
10	SQL Keys database	SQL
11	SQL Ontologies Database	SQL
12	Reliability Configuration	Conf
13	Auto scaling Configuration	Conf

4.3 Repository of Service Descriptions (RSDs)

This repository is a set of documents that describe the composition and configuration of the TRENCADIS services. Each document specifies the set of VMs

involved in a service, and for each type of VM, its hardware requirements (number of CPUs, memory, etc.), software components, indicating the configuration required to set up the services (creating users, directories, deploy Grid services, create SQL databases, install Java APIs etc..). To write the documents, the Resource Application Description Language for Cloud environments (RADL) [22] has been used. RADL expresses in a simple and declarative way the requirements of the TRENCADIS services on a set of VMs, as well as to obtain information from the VMs already instantiated. A RADL document consists of three sections: The first one (system) declares the requirements of different types of VMs required. The second one (configuration) describes the configuration steps required for each type of VMs. The last one (deploy) indicates the number of instances of each one.

Table 3. Created RADLs for TRENCADIS Infrastructure

ID	RADL	ID VMI	ID Installation File
1	DICOM Storage Service	[1,3 or 4 or 5,6 or 7]	[1,6,7]
2	Key Server Service	[1,3]	[2,8,10]
3	EOUID Generator Service	1	[3]
4	Storage Broker Service	1	[4]
5	Ontologies Server Service	[1,3]	[5,9]
6	VOMS Service	2	–
7	Information Server Service	1	–
8	TRENCADIS Center	[1,2,3]	[3,4,5,9]

Table 3 lists RADL documents created in this work, one for each TRENCADIS service. A RADL document has also has been created to describe all services grouped by TRENCADIS center.

As an example, figure 3 shows the RADL document needed to deploy the Key Server service where two VMs are needed (GT4-JAVA-ANT and POSTGRESQL). The first one needs a Scientific Linux (SL) 5.7 VM with Globus Toolkit 4, java and ant installed. The second one also needs a SL 5.7 VM but only with PostgreSQL installed. Also two network interfaces are specified: a private network to connect the instances and a public network interface that is used by the Grid service for interacting with other TRENCADIS services.

4.4 Infraestructure Instantiator (II)

This component is in charge of interpreting the infrastructure descriptions specified in the RADL documents, to finally perform the effective deployment of the instances of the selected VMI in a cloud environment or in a virtualization system. It is implemented by means of the Infrastructure Manager (IM) [22]. The IM is a service that provides a high level REST API and Web Interface to enable the deployment and automatic contextualization of Cloud infrastructures. It provides a set of functions to create and destroy virtual infrastructures and also to provision and relinquish computational resources in an elastic manner. The IM enable

```
network private
network public (outbound = 'yes')

system GT4_JAVA_ANT (
   cpu.arch='x86_64' and cpu.count>=1 and memory.size>=1024m and
   net_interfaces.count = 2 and net_interface.0.connection = 'public' and
   net_interface.1.connection = 'private' and
   disk.0.os.name='linux' and disk.0.os.flavour='Scientific Linux' and
   disk.0.os.version='5.7' and
   disk.0.application contains (name='globus', version='4') and
   disk.0.application contains (name='java', version='1.6.0.35') and
   disk.0.application contains (name='ant', version='1.8.2')
)
system BACKEND_POSTGRESQL (
   cpu.arch='x86_64' and cpu.count>=1 and memory.size>=1024m and
   net_interfaces.count = 1 and net_interface.0.connection='private' and
   disk.0.os.name='linux' and disk.0.os.flavour='Scientific Linux' and
   disk.0.os.version='5.7' and
   disk.0.application contains (name='PostgreSQL', version>='8.4.9')
)

configure GT4-JAVA-ANT (
     add_user {'trencadis': }
     deploy_gs {'Key_Server_Grid_Service.gar': }
     install_conf_API_SQL_Keys_Database
                            {'TRENCADIS_Java_API_SQL_Keys_DB.jar': }
)

configure BACKEND_POSTGRESQL(
     add_user {'trencadis': }
     create_database {'SQL_Keys':
                         file =>'database.SQL'}
)

deploy GT4_JAVA_ANT 1
deploy POSTGRESQL 1
```

Fig. 3. RADL document for Key Server service.

to connect with different Cloud systems such as OpenNebula [24], OpenStack [25] and Amazon EC2. It also enables to connect with virtualization systems like KVM [26] using the LibVirt interface [23]. The functional scheme is the following: The first step is to connect to the VMRC to select the most suitable image(s) with respect to the user requirements. Then it selects the user credentials to access the cloud deployments or virtualization systems to launch the VM instances. To launch them, it must translate the RADL requisites into instances of the selected system. Finally it waits the VMs to be running in order to perform the contextualization process using Puppet.

For example, for deploying the Key Server service, the IM connects to the VMRC to select the VMI 1 and 3. Then, it submits the VM instances to a cloud or virtualized infrastructure. Finally, it waits for the VMs to be running in order to perform the contextualization using the software components with ID 2, 8 and 10.

4.5 Virtual Machine Manager (VMM)

This component is in charge of managing the instances of VMs launched by the IM. It must provide functionality to create and destroy infrastructures and also data persistence when undeploying VMs. This component can be implemented in

the platform using software such as OpenStack or OpenNebula or public clouds such as Amazon EC2.

5 Test Deployment

To test the platform designed in this work, a complete TRENCADIS infrastructure has been set up following the deployment model presented in section 3. This infrastructure is shown in figure 4 deployment has emulated all the services required for three medical centers and a TRENCADIS center.

For managing the instances of VMIs required to deploy the TRENCADIS services, a public cloud, a private cloud and a virtualization platform have been combined. Figure 4 shows the VMs involved in the deployment and the VMM used in each one.

In particular, the private cloud used has been supported by four Dell Servers in Blade format (M600 and M610 models). Each server has eight cores and 16 Gb of RAM and are mounted on a M1000e chassis. Three nodes were available to run VMs submitted by OpenNebula (version 3.4.1) while one node could run VMs submitted through OpenStack (Essex release). The public cloud used has been Amazon EC2 and the virtualization platform has been supported by the KVM hypervisor. The services Ontologies Grid service and EOUID Generator service have been deployed using the reliability configuration file, setting two instances in Amazon EC2 and the corresponding load balancer for proper workload distribution. The service Storage Broker Grid service has been deployed using the auto scalable configuration file, setting an autoscaling group of Amazon EC2, where a new storage a new Storage Broker is deployed if the CPU Utilization is above 80% for more than 1 minute. The Amazon EC2 API supports auto scaling and allows creating groups of instances, maintaining a minimum and maximum of instances actives, automating the creation of new replicas to replace failed ones [27].

6 Conclusions and Future Work

The platform designed for deployment TRENCADIS infrastructures enable to combine virtualization technologies and clouds (public and private), depending on the needs of the deployment.

The use of these technologies significantly simplifies the deployment, while improving its performance and reliability, as these technologies enable deploying new services provisioning on demand, in an elastic and dynamic way.

Furthermore, the platform allows you to create, deploy and configure on-demand services, combining different versions according to the required needs, efficiently and without losing its scalability.

In this work, we have defined a methodology that has allowed to identify software components required to deploy an infrastructure TRENCADIS, analyzing all its services. Based on the identified components, these have been implemented in the platform as VMIs or configuration files. This methodology can be applied to other infrastructures and the platform can be used in other areas different

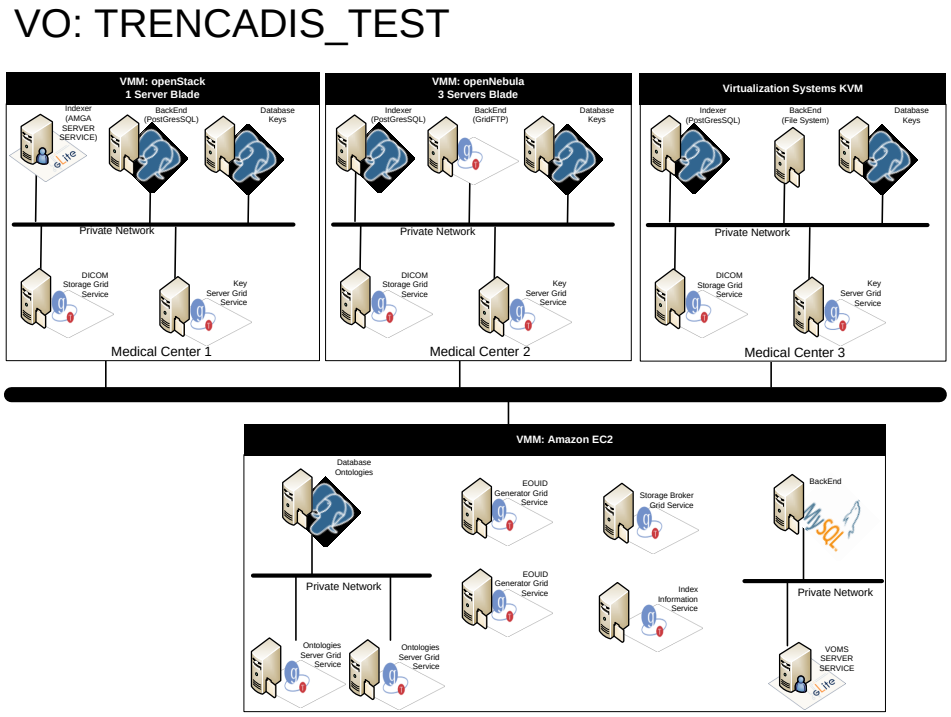

Fig. 4. Test Deployment of a TRENCADIS infraestructure

from TRENCADIS. Therefore, as a future work we plan to identify and apply the methodology in use cases different from TRENCADIS.

Acknowledgements

The authors wish to thank the financial support received from The Vicerectorat d'Investigació de la Universitat Politècnica de València (UPV) to develop the project "Diseño de Componentes Cloud Facilitadores del Despliegue y la Alta Disponibilidad de Servicios TRENCADIS, para compartir Imágenes Médicas DICOM e informes Asociados DICOM-SR", with reference 20111013.

References

1. C. Vázquez, E. Huedo, R.S. Montero, I.M. Llorente, *"On the use of clouds for grid resource provisioning"*, Future Generation Computer Systems, 27(5): 600-605, 2011
2. A.M. Kuo, *"Opportunities and Challenges of Cloud Computing to Improve Health Care Services"*, J Med Internet Res. 13(3): e67, 2011
3. European Commission: *"Safeguarding privacy in a connected world a European data protection framework for the 21st century"*. Tech. rep., European Commission, Brussels, Belgium (2012)

4. J. Kommeri, M. Niinimki, H. Mller, *"Safe storage and multi-modal search for medical images"*, Stud Health Technol Inform, 169:450-454, 2011
5. I. Blanquer, V. Hernńdez, F.J. Meseguer, J.D. Segrelles,*"Content-based organisation of virtual repositories of DICOM objects"*, Future Gener. Comput. Syst., 25(6):627637 (2009)
6. Medical Imaging & Technology Alliance, *"DICOM - Digital Imaging and Communications in Medicine"*, http://medical.nema.org
7. D. Segrelles, I. Blanquer, J. Salavert, V. Hernandez, J. Franco, G. Diaz, R. Ramos, R. Medina, L. Marti, M. Guevara, N. GonzÍez, J. Loureiro, I. Ramos, *"Exchanging Data for Breast Cancer Diagnosis on Heterogeneous Grid Platforms"*, Computing and Informatics. 31(1): 3-15, 2012
8. B. Narasimhan, R. Nichols, *"State of Cloud Applications and Platforms: The Cloud Adopters' View"*, Computer, 44(3): 24-28, 2011
9. *"Puppet:"* http://puppetlabs.com . visited 16/04/2013
10. *Chef*: http://www.opscode.com/chef/. visited 16/04/2013
11. *Amazon Inc. Amazon Elastic Compute Cloud (Amazon EC2)*. http://aws.amazon.com/ec2. visited 30/10/2012
12. *Apache Whirr*: http://whirr.apache.org/. visited 16/04/2013
13. *CloudFoundry Open Source*: http://www.cloudfoundry.org/. visited 16/04/2013
14. N. Chohan, C. Bunch, S. Pang, C. Krintz, N. Mostafa, S. Soman, and R. Wolski. *AppScale: Scalable and Open AppEngine Application Development and Deployment.* In International Conference on Cloud Computing, Oct. 2009
15. *Google App Engine*: http://cloud.google.com/appengine. visited 30/10/2012
16. B. Koblitz, N. Santos, V.Pose. *"The AMGA Metadata Service"*. JGC. ISSN: 1570-7873 (Print) 1572-9814 (Online). Volume 6, Number 1 / March de 2008. Pages 61-76. Springer Netherlands, 2007.
17. I. Blanquer, V. Hernandez, J. Salavert, D. Segrelles. *"Integrating TRENCADIS Components in gLite to Share DICOM Medical Images and Structured Reports"*. HealthGrid Conference, 2010, pp 64-75 .ISBN 978-1-60750-582-2.
18. *"The Worlds Leading Graph Database"*. http://neo4j.org . Visited 16/04/2013.
19. *"Cloud Data Management Interface - Specification Version 1.0.2h"*, SINA, 2010. http://cdmi.sniacloud.com . visited 26/10/2012
20. *"gLite-Lightweight Middleware for Grid Computing"*. http://glite.cern.ch . visited on 16/04/2013
21. Carrion, Jose V., Germn Molto, Carlos De Alfonso, Miguel Caballer, and Vicente Hernandez. 2010. *"A Generic Catalog and Repository Service for Virtual Machine Images."* In 2nd International ICST Conference on Cloud Computing (CloudComp 2010).
22. C. de Alfonso, M. Caballer, F. Alvarruiz, G. Molto, V. Hernandez, Infrastructure Deployment Over the Cloud, in: 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011), 2011, pp. 517-521.
23. Bolte M, Sievers M, Birkenheuer G, Niehrster O, Brinkmann A. *"Non-intrusive virtualization management using libvirt"*. Proceedings of the 2010 Conference on Design, Automation and Test in Europe (DATE), Dresden, Germany, 2010.
24. J. Fontan, et al. *"OpenNebula: The Open Source Virtual Machine Manager for Cluster Computing"*. In Open Source Grid and Cluster Software Conference, May 2008.
25. *OpenStack Open Source Cloud Computing Software*. http://www.openstack.org, visited 16/04/2013
26. Kivity, A., Kamay, Y., Laor, D., Lublin, U., and Liguori, A. KVM: the linux virtual machine monitor. In Ottawa Linux Symposium (July 2007), pp. 225230.
27. http://aws.amazon.com/developertools/2535 . visited on 16/04/2013