A Platform to Deploy Customized Scientific Virtual Infrastructures on the Cloud

Miguel Caballer*, Damián Segrelles, Germán Moltó, Ignacio Blanquer

Instituto de Instrumentación para Imagen Molecular (I3M). Centro mixto CSIC – Universitat Politècnica de València – CIEMAT Camino de Vera s/n, 46022 Valencia, Spain.

SUMMARY

This paper presents a software platform to dynamically deploy complex scientific virtual computing infrastructures, on top of Infrastructure as a Service (IaaS) Clouds. The platform orchestrates different services to provision the virtual computing resources. It dynamically installs the appropriate software to satisfy the requirements of a researcher, both on public and on-premise Clouds. The platform provides a web interface to enable the users to easily management of the lifecycle of virtual infrastructures. It enables users to define infrastructures, share them with other users, deploy and relinquish them, add or remove resources dynamically, create and share application recipes, etc. The paper also describes three case studies to deploy complex infrastructures, namely a Hadoop cluster, a single-node to perform NGS sequencing and a gateway for users to access the European Grid Infrastructure (EGI). This platform promotes a better use of on-premise hardware resources of a research center by allocating the computing resources just-in-time to the specific life time of the virtual infrastructures as well as the deployment of the very same infrastructures on a public Cloud.

Received ...

KEY WORDS: cloud computing, virtual infrastructures

1. INTRODUCTION

The diverse computing requirements for scientific applications require complex hardware infrastructure configurations and potentially incompatible specific software requirements. In a multi-disciplinary environment different researchers typically share the same hardware. Therefore, the adequate provision and configuration of computing resources could be unaffordable or impractical due to the technical overhead of switching among configurations and the effort on the configuration and customization of the infrastructures. Under these circumstances, the use of frameworks to automate the deployment and configuration of virtual computing infrastructures is necessary. Furthermore, to reduce the carbon footprint, it is especially important to properly and efficiently manage the computing resources of a research center so that the level of service and versatility is maintained without requiring additional investments in hardware.

For that purpose, Cloud computing [1][1, 2] is a paradigm to rapidly provision ICT resources, mainly computing and storage, that can be customized and configured to fit a particular research activity. This enables to dynamic deployment of virtual infrastructures on top of a fleet of virtual machines running on a physical hardware, when using the Infrastructure as a Service (IaaS) Cloud service model. The usage of virtualization [3] enables hardware usage to be increased thus reducing the investments on additional hardware. In the case of a public Cloud provider, such as Amazon Web

Services (AWS)*, Microsoft Azure† or Google Cloud Platform‡, a pay-per-use model is employed so that users are charged for the resources consumed in terms of hours of computing, network traffic, etc. In the case of on-premise Clouds, tools such as OpenStack [4], OpenNebula [5] or Eucalyptus [6] enable Cloud infrastructures to be deployed on top of the organization's computing hardware.

In this paper, a general platform to deploy on demand customizable virtual computing infrastructures is presented, with the precise software configuration required to perform specific research activities. This platform is able to integrate computing resources both from public and on-premise Clouds. This introduces an unprecedented degree of flexibility when compared to managing physical infrastructures to support the computing requirements of complex computational research activities. This platform enables the deployment of customised on-demand virtual infrastructure to be used as the computational backend for different scientific applications as well as a scientific gateway for users to access larger distributed computational infrastructures for eScience.

The remainder of the paper is structured as follows. First, section 2 presents the novelty of our platform with respect to other existent tools in the area of dynamic deployment of virtual infrastructures. Next, section 3 describes the software architecture of the platform together with the underlying components employed. Then, section 4 describes the usage of this platform to deploy virtual infrastructures to support different use cases. Finally, section 5 summarises the benefits and drawbacks of the proposed platform and points to future work.

2. RELATED WORK

There exist works in the literature and software tools that address the deployment of virtual infrastructures. Several cloud providers, such as AWS provide tools to deploy virtual infrastructures. In particular, CloudFormation [7] and OpsWorks [8] provides developers and systems administrators with an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion.

The Nimbus project team group has developed a set of tools to deploy virtual infrastructures: the Context Broker [9], the Recontextualization Broker [10, 11] and cloudinit.d [12]. In particular, the last tool submits, controls and monitors Cloud applications. It automates the VM creation process, the contextualization and the coordination of service deployment. It supports multiple clouds and the synchronization of different "runlevels" to launch services in a defined order. Furthermore it provides a system to monitor the services that uses user-created scripts to ensure that they are running. This system checks for service errors, re-launching failed services or launching new VMs. However, it enables the contextualization of VMs using simple scripts, which are insufficient in complex scenarios with multiple VMs and different Operating Systems.

Another example is Apache Whirr [13]. It supports deploying clusters both on EC2 and Rackspace. The user specifies the number of instances needed and the roles they must provide. The Virtual Machine Images (VMIs) to be used have to be specified beforehand. It has been initially designed to launch Hadoop clusters, but it can be extended adding new Java classes to implement the installation and contextualization of the new roles defined. It does not enable elasticity management, so once the cluster is launched its size cannot be modified by the user.

Other interesting tool is Wrangler [14]. It enables the users to define their requirements using an XML file, and specify the scripts to be used to configure the VMs to adopt a specific role within the virtual infrastructure. In this case the scripts are stored in the wrangler coordinator node, so it does not need to be previously copied in the VMIs. But the VMIs require a prior step to prepare them by installing the wrangler agent and configuring it to connect to the coordinator node.

Currently there are also some initiatives from standardization organizations, such as OASIS with the Topology and Orchestration Specification for Cloud Applications (TOSCA) [15] to describe

*http://aws.amazon.com
†http://azure.microsoft.com
‡https://cloud.google.com

service templates across *aaS (SaaS, PaaS, IaaS) [1] layers and connecting them to the resource abstraction layer. It enables the description of a service with a high level topology and plan for implementation and configuration. A service specified with TOSCA typically describes virtual hardware, software, configuration, and policies necessary to orchestrate the service. Currently OpenTOSCA [16] is the reference implementation. Although this specification considers the contextualization of the VMs it does it using executable files, and not using some high level contextualization language.

Finally, SixSq SlipStream* provides a web portal to deploy and configure a set of VMs. The configuration of the nodes are made by means of a list of packages to install and a script file that can be executed in each VM, which can be parameterized to create more functional and customized scripts. It can access a large list of Cloud platforms, both public and on-premise: EC2, Azure, OpenNebula, OpenStack, OCCI, etc. The main limitation is the static selection of VMIs. In addition, it does not enable elasticity management and, therefore, once the infrastructure is deployed its size cannot be modified by the user.

One common limitation of all the previously analysed systems is the usage of manually selected base images to launch the VMs. This is an important limitation because it implies that users must create their own images or they must previously know the details about software and configuration of the image selected. This limitation affects the reutilization of the previously created VMIs, forcing the user to waste time testing the existing images or creating new ones. Another issue is that most of them need to use a VMI specifically configured to support their environment, requiring a specific software installed or a set of scripts prepared. Another important limitation is the usage of simple scripts in the contextualization, instead of DevOps tools such as Puppet [17], Chef [18], or Ansible [19], which create (nearly) system-independent configurations. Only the Nimbus "Recontextualization Broker" uses Chef to perform these tasks. This problem is exacerbated in some tools where the configuration scripts must also be stored in the base image of the VM. Furthermore, most of the aforementioned tools do not provide an easy language or an user friendly interface, hindering the usage for non advanced users.

The platform defined proposes an architecture which separates the management of VMIs, the infrastructure descriptions and their installation and deployment on a Cloud. It offers a degree of flexibility that the previous aforementioned works do not offer, allowing to extend and improving the related works presented in the following aspects:

- The platform does not exclusively focus on a specific type of infrastructure (PC cluster, grid testbed, parallel environment, etc), since it allows to deploy any kind of complex infrastructure.
- One common limitation of the previous works is the usage of pre-packaged VMIs. This is an important limitation because users are limited to use the pre-selected software of the VMIs or they must install the desired software after the virtual infrastructure has been provisioned (leading to potential software conflicts). This platform, instead, provides the ability to automatically install at runtime software and data dependences in the virtual infrastructure, and perform complex configurations of the infrastructures. Therefore, it is easier to keep the infrastructure updated, since new version of packages can be installed in each new deployment.
- The platform enables users to reuse both the infrastructure descriptions, using a repository of high level infrastructure descriptions, and the VMIs, using a catalog with their specific metadata.
- The same complex infrastructure can be deployed both on-premise and in a public Cloud, thus enabling to outsource computation to the public Cloud when there is a shortage of computational resources on-premise, even creating hybrid infrastructures that harness resources from different Cloud providers.

*https://slipstream.sixsq.com

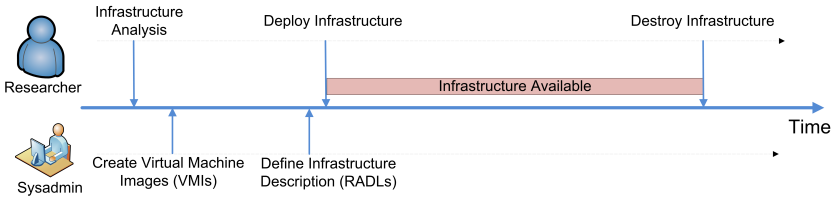

Figure 1. Time line and main actors involved in the virtual computing infrastructures.

3. PROPOSED ARCHITECTURE

In the scenario proposed in this paper it is assumed that a research institution has a pool of computing hardware that is managed by a Cloud Management Platform (CMP), such as OpenNebula, in order to run virtual computing infrastructures on top of that hardware. This provides all the benefits of virtualization such as isolation and better hardware utilization, thus avoiding the usage of dedicated hardware. Many organizations are already using virtualization to leverage server consolidation [20] so that, for example, a single machine hosts different virtual machines (VMs) with separate roles such as mail server, application server, etc.

Alternatively, the scenario also considers a research center that outsources part of their computing to a public Cloud provider, in order to provision VMs for operational support of their staff. Ultimately, scientists could certainly bypass their research institution and deploy the required resources on a public Cloud by themselves, if no support is provided by their institution.

Under the aforementioned assumptions, the presented platform enables creating virtual computing infrastructures on top of public and on-premise Clouds in order to support research activities. Therefore, the main aim of the platform is to simplify the definition of virtual computing infrastructures and to perform their automated deployment, by dynamically provisioning and relinquishing the related resources, together with the appropriate configuration to fit the requirements of a certain scientific application.

Figure 1 summarises the main steps involved when interacting with the platform. First of all, the researcher must analyse the infrastructure requirements for a specific application, in terms of hardware, software, and special configuration requirements. For example, a researcher could require a Hadoop cluster with N 64-bit virtual machines with Ubuntu GNU/Linux 12.10. The researcher passes the requirements to a system administrator (*sysadmin*) of the platform, which could be the researcher itself if he has the appropriate skills. The sysadmin must check if there are appropriate Virtual Machine Images (VMIs) stored in the repository and if not, he must install and configure them. Finally he must register the new VMIs in the repository and catalog service including the correct metadata (OS, applications, etc.) to enable its usage in the platform.

The sysadmin must distinguish between the software that can be dynamically deployed at runtime on the VMs and the one that should be pre-installed in the VMIs due to complex software requirements or a lengthy installation process. Then, the sysadmin describes the infrastructures using a higher level declarative language called RADL, which will be explained later on. Each description is a recipe of the hardware, software and configuration required to instantiate a given infrastructure, regardless of the underlying Cloud infrastructure to be employed. An infrastructure can be composed of one or more VMs with possibly different configurations. The usage of a high level recipe means that the same computing infrastructure can be deployed on an on-premise Cloud and a public Cloud, in case no spare hardware resources are available in the on-premise Cloud. Also, to have recipes for each infrastructure enables the automation of their deployment process for different activities, such as a periodic task.

The infrastructure descriptions are made available to the researchers through a web application. Infrastructures can therefore be instantiated, which means to actually create and configure the VMs to deploy the computing infrastructure in a Cloud back-end. This triggers all the work of the platform which is in charge of deploying the VMs out of the VMIs, by interacting with different IaaS Cloud providers, dynamically installing the specified software and providing the configuration among the

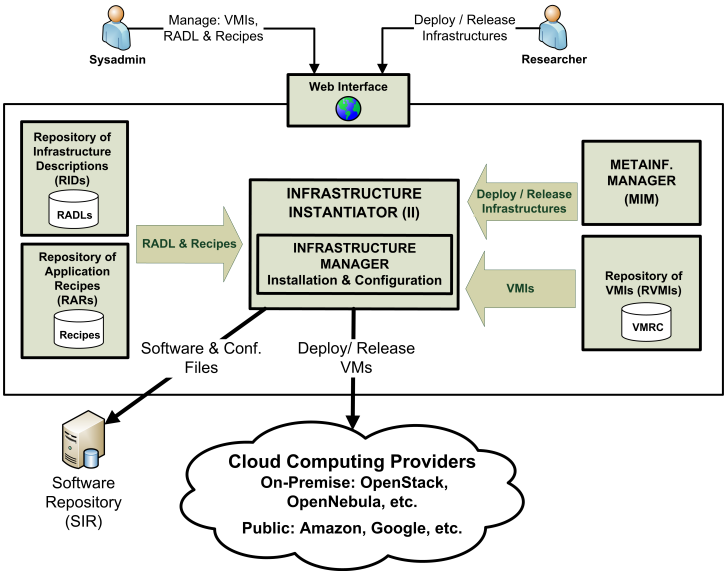

Figure 2. Main architecture of the platform.

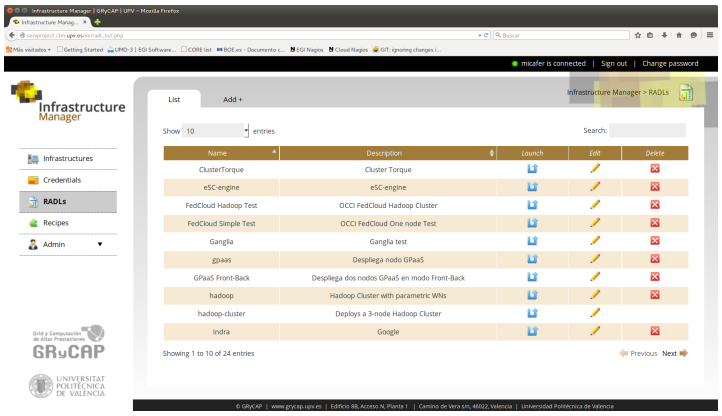

Figure 3. Web User Interface to the platform

VMs in order to create an available virtual computing infrastructure that satisfies the requirements specified in the high level, declarative RADL description.

Once the virtual computing infrastructure is up and running the researcher can access it via SSH or using remote desktop tools such as FreeNX, depending on the configuration specified. She will be also in charge of destroying the infrastructures in order to release the provisioned hardware resources.

Figure 2 shows the architecture of the platform. The central component is the Infrastructure Instantiator (II) that is in charge of deploying and contextualizing the VMs of the virtual infrastructures. The II uses the infrastructure descriptions, which are RADL documents stored in the Repository of Infrastructure Descriptions (RIDs). These RADL documents can reuse the application recipes stored in the Repository of Application Recipes (RARs). It also selects the most suitable VMIs available from the Repository of Virtual Machine Images (RVMIs). Then, in the contextualization step, it uses the software and data packages from the software repositories. Finally the MetaInfrastructure Manager (MIM) controls the II by organizing and scheduling when the infrastructures will be deployed or released. The following subsections describe the main components and technologies of the platform.

3.1. Repository of Virtual Machine Images (RVMIs)

A VMI represents a bundle of the installation of an Operating System together with a set of applications, configuration and data. These base VMIs can later be modified at runtime to install the specific software applications and/or data required for a given educational activity. Therefore, a set of base VMIs are expected to be used, from which new installations can be performed, in order to create additional VMIs. For example, one could think of a VMI based on CentOS GNU/Linux, which can be modified at a later stage to install an application software, such as Octave, to satisfy the requirements of a given practical lesson. This situation increases the number of VMIs, thus requiring proper management in the shape of a Repository of Virtual Machine Images (RVMIs). This component allows reusing the VMIs so that a reduced number of them is employed. This has, among other advantages, to ease keeping up-to-date VMIs with the latest software updates.

The RVMIs has been implemented by means of the Virtual Machine image Repository & Catalog (VMRC) software [21]. This software enables the storage and indexing of VMIs together with metadata concerning the OS, hypervisor, disk size and applications installed (name, version and location) among other. VMRC supports a query language in order to obtain a ranked list of VMIs that satisfy a given set of requirements. This enables the platform to query the VMRC, for example, for a VMI with GNU/Linux Ubuntu version greater or equal to 12.04, with OpenJDK JRE 6 installed. This is an open source component*.

3.2. Infrastructure Instantiator (II)

The Infrastructure Instantiator (II) is the component in charge of managing all the life-cycle of virtual infrastructures. It instantiates a RADL document to create the infrastructure by orchestrating all the required tasks. It can add or remove nodes dynamically to fit the size of the infrastructure to the real requirements, and apply the required configuration of the software components. It has been implemented using the Infrastructure Manager (IM) [22, 23].

The II uses the RVMIs to get the most suitable VMI considering the restrictions defined in the RADL document. Then it contacts the Cloud back-end to provision the specified VMs and, finally, it uses Ansible to configure the whole infrastructure. The II has a modular design enabling the interoperability with several cloud deployments. It currently supports on-premise Cloud tools such as OpenNebula and OpenStack as well as any OCCI-compliant Cloud provider. It can also interact with Amazon Elastic Compute Cloud (EC2) and Google Compute Engine (GCE) to deploy VMs on that public Cloud providers. Furthermore, it can also deploy Docker containers † instead of VMs on an appropriate hardware infrastructure.

In the contextualization phase (installation and configuration), all the software and data needed to fulfil the requirements of the RADL document are dynamically retrieved from the Software Repository (SR). The IM is implemented as a service that exports a simple interface that can be used from different client programs. Two client interfaces have been developed, a simple command line program and an API (available through XML-RPC and REST).

3.3. Repository of Infrastructures Descriptions (RIDs)

To describe an infrastructure, both the hardware characteristics and the software and configuration issues should be specified. All these features must be described in a formal document to enable the automation of these tasks. These documents are expressed using the Resource Application Description Language (RADL) [22, 23], which is a simple high level and declarative Domain Specific Language (DSL) to describe the hardware and software features of a virtual infrastructure (see an example in Figure 5). A RADL document specifies the requirements of each type of VM needed in the infrastructure and the number of instances of each type to be deployed on top of the IaaS Cloud.

*Available at http://www.grycap.upv.es/vmrc
†https://www.docker.com/

```
system node (
memory.size>=2g and
disk.0.os.flavour='ubuntu' and
disk.0.os.version>='12.04' and
disk.0.applications contains (name='HMMER') and
disk.0.applications contains (name='MAFFT')
)
deploy node 1
```

Figure 4. RADL document that used the recipes of HMMER and MAFFT.

The Repository of Infrastructures Descriptions (RIDs) stores a set of RADL documents in a web application where the user can specify a name and a description for each document. The creator of an RADL can also specify a set of permissions to access the document, enabling to share the RADLs with different groups of users with different permissions (read, write, deploy). This represents a central access point to share RADL documents created by the sysadmins that can be used by the researchers using a simple web interface (as shown in Figure 3). This way, infrastructures can easily be deployed by only means of a web browser, without any special client-side configuration.

The RADLs can further be customised via some parameters that can be defined at deploy time. In this case, the user that creates the infrastructure can specify some specific values of the infrastructure without the modification of the RADL. Any value of the RADL can be specified as a parameter and they enable the end user to specify, for example, the number of nodes of a cluster to be launched, the amount of memory, etc. For that, a special notation (@*input.*<*var_name*>@) is employed.

3.4. Repository of Application Recipes (RARs)

The Repository of Application Recipes (RARs) enables the sysadmin to store a set of recipes with all the steps needed to install an specific application. It enables the creation of a repository of well known applications that users can specify in the RADL definition. Therefore, non advanced users can define complex contextualization steps with a very simple RADL. It is also accessed using the Web user interface of the platform shown in Figure 3.

For example the sysadmin has defined the recipes for the applications HMMER* and MAFFT†. In that case, users that want to deploy nodes with those tools can compose their RADLs to reference the aforementioned recipes. For example in Figure 4 the user specifies a node with 2GB of RAM, with an Ubuntu 12.04+ with both applications installed. This greatly simplifies the specification of complex applications to be dynamically installed on the nodes of a virtual infrastructure.

3.5. MetaInfrastructure Manager (MIM)

The MetaInfrastructure Manager (MIM) organizes and schedules when the infrastructures will be used, deploying or releasing the virtual infrastructures defined, in an automatic or manual way. This component is the bridge between the II and the researcher. The MIM interacts with the researcher through a web interface, and launches instructions to the II via its API. The web interface is also integrated with the RID. Therefore, the same interface is employed to access the RADL documents and deploy the infrastructures (Figure 3), enabling the users to provision customised virtual infrastructure with a single click.

Virtual infrastructures can be dynamically relinquished when they are no longer used thus cutting down the costs. These costs are energetic, in the case of on-premise Clouds or economic, in the case of public Clouds. Therefore, a cost-effective platform to provision computational capacity is provided.

*http://hmmer.janelia.org/
†http://mafft.cbrc.jp/alignment/software/

It also enable the users to schedule lifetime of the infrastructures. The user can specify to the platform the dates and times that the infrastructure will be deployed and terminated. For example it can be used to launch an infrastructure at 9:00 every working day and terminate in the afternoon at 20:00 when the workday is finished. In this way the users can access the infrastructure all the working days but the infrastructure is not consuming resources when it is not used, thus reducing the costs.

4. USE CASES

This section shows the advantages of the proposed platform for three different real scientific applications with specific computing and configuration requirements. In particular, two different Cloud platforms have been used in the following tests: The first is an on-premise cloud with OpenNebula composed of three Dell blade nodes (M600 and M610), each one with two Quad-Core processors and 16 GB of RAM. The second one uses the public Cloud infrastructure provided by Amazon Web Services.

The tests show the time needed to deploy the infrastructures. This time includes all the steps required to deploy the VMs, install Ansible in one node (designated as master node), then use it to contextualize all the deployed VMs with the user requirements expressed in the RADL document.

4.1. Use Case: Hadoop cluster for Hadoop BLAST

MapReduce [24] is a programming model widely used to process large amounts of data. MapReduce is implemented on top of Hadoop [25] which provides the implementation of a HDFS replicated and distributed file system to manage large volumes of data. Hadoop and MapReduce are widely used in Cloud computing infrastructures to tackle Big Data problems. MapReduce is appropriate for problems of large text indexing and searching, such as the genome analysis in bioinformatics. The problem of alignment of sequences of nucleotides using BLAST (Basic Local Alignment Search Tool) [26], has been selected, as one of the most popular tools in homology search and function analysis in biomedicine. BLAST was used in conjunction with other software developed to enable this application to run in a Hadoop cluster*.

The IM enables to use Ansible Galaxy† roles in the configuration sections. Only a special application must be specified in the system definition with the name *"ansible.modules.[user].[name of the role]"*. In the example shown in Figure 5 the role hadoop of the user micafer is specified. Then, in the configure sections, the roles specified can be used to configure the infrastructure. This enables the configuration of a Hadoop cluster for non advanced users. This recipe deploys and configures a fully operational Hadoop cluster. Notice that the number of nodes is defined as an input value to the RADL thus enabling the user to specify it at deployment time.

Table I shows the time spent to deploy several Hadoop clusters with different configurations in both AWS (using the Amazon EC2 service) and OpenNebula (ONE). The size of the clusters was 6, 11 and 51 nodes. One was the front-end and the rest were Working Nodes (WNs). The table includes the time concerning the *deployment* (provisioning and configuration of the infrastructure), the addition of a new node to the cluster and, finally, the removal of a node from the cluster. Notice that modifying the number of nodes typically requires a reconfiguration of the whole infrastructure to propagate the new state.

In EC2, the instance type selected by the IM was "m1.small", since the only requirement specified in the RADL is to have more than 1.5 GB of RAM. For the front-end node, IM selected a "c1.medium" as it is the cheapest one with two cores and 1.5 GB of RAM. The *us-east-1* region (located in North Virginia, USA) was employed.

*http://salsahpc.indiana.edu/tutorial/hadoopblast.html
†https://galaxy.ansible.com/

```
network privada
network publica (outbound = 'yes')

system front (
cpu.arch='x86_64' and cpu.count>=2 and
memory.size>=1536m and
net_interface.0.connection = 'public' and
net_interface.0.dns_name = 'hadoopmaster' and
net_interface.1.connection = 'private' and
disk.0.os.name='linux' and
disk.0.os.flavour='ubuntu' and
disk.0.os.version='12.04' and
disk.0.applications contains (name='ansible.modules.micafer.hadoop')
)

system wn (
cpu.arch='x86_64' and
memory.size>=1536m and
net_interface.0.connection='privada' and
disk.0.os.name='linux' and
disk.0.os.flavour='ubuntu' and
disk.0.os.version='12.04'
)

configure front (
@begin
---
 - roles:
    - { role: 'micafer.hadoop', hadoop_master: 'hadoopmaster', hadoop_type_of_node: 'master' }

@end
)

configure wn (
@begin
---
 - roles:
    - { role: 'micafer.hadoop', hadoop_master: 'hadoopmaster' }

@end
)

deploy front 1
deploy wn @input.WNs@
```

Figure 5. RADL document of the Hadoop Use Case.

Table I. Haddop Cluster creation times

	6 node (ONE)	11 nodes (ONE)	11 nodes (EC2)	51 nodes (EC2)
Deployment	12:05	22:13	9:29	14:26
Node Addition	5:13	5:33	5:46	11:23
Node Removal	1:18	1:32	3:17	6:36

In OpenNebula, deployment times increases with the number of VMs, since the on-premise Cloud platform suffers from a bottleneck in the network and disk access. This effect does not show up in EC2, in which the deployment of the clusters takes considerable lower time.

Concerning the dynamic modification of the clusters, it can be seen that this takes longer in EC2, since provisioning and specially configuring the VMs require multiple connections from the IM service, which is located in our on-premise Cloud (in Valencia, Spain), to the deployed VMs in EC2. Notice that the on-premise OpenNebula Cloud is in the same local network as the IM service, whereas the VMs are deployed in a geographically distant region. If necessary, these steps can be sped up by deploying the IM within the same AWS region.

4.2. Use Case: NGS: Next Generation Sequencing

The analysis of mutations consists on identifying significant variants of an individual's genome with respect to the consensus reference genome and the existing known mutation databases. With the advent of massive sequencing (also known as Next Generation Sequencing - NGS), this process has increased in complexity and resource requirements. There are a wide range of tools used by the researchers for the preprocessing, alignment to reference, identification and variants and searching of variants in data bases which constitute well-known processing pipelines. Those tools require local copies of the reference genome and the individual's sequences.

The use case implemented integrates FastQC* for quality analysis, Bowtie2† for the alignment, Samtools‡ for sorting, data conversion and pile-uping, GATK§ for the computation of Variant Calls and Galaxy¶ as interface.

This use case shows how to integrate a set of the necessary tools in a VM to provide a scientist with the very specific environment to carry out her research. By properly defining the recipes the users are guaranteed to access a specific version of the tools. This increases repeatability and could enable reviewers of scientific publications to deploy the very same environment to verify the results discussed in an article. The use of the IM service is free and the deployment can be done in any public Cloud or on-premise infrastructure.

NGS analysis may have different requirements in terms of the number of cores and memory at different steps of the processing pipeline. This approach may be used to use local on-premise resources as much as possible and to scale out to public larger resources when instances with larger needs are required. For example, 64 GB instances may not be available on a local infrastructure but they may not be necessary for all the experiments, only for large genomes or special configurations.

Creation time of a single node has been of 11:11 in ONE and 10:42 in AWS EC2. In this case some computational processing task has been added in the configuration step. In particular the experiment creates indexes for the chromosomes 7, 8 and 9 of the chicken reference genome from UCSC‖ (319MB file) that takes about 3-4 minutes. This step is included as it is a basic initial step in the analysis of mutations.

4.3. Use Case: EMI-UI

A representative use case is the setup of a User Interface (UI) based on the EMI-3** distribution so that users can access the European Grid Infrastructure (EGI)††, a publicly funded e-infrastructure to give scientists access to more than 370,000 logical CPUs, 170 PB of disk capacity across Europe.

This use case consists of deploying and configuring an EMI-3 User Interface. User accounts can be created so that users connect via SSH to that instance in order to submit their jobs to be executed on the EGI.

Creation time has been of 35:40 in ONE and 35:20 in AWS EC2. The creation time of this node is relatively large due to the large number of packages to install in the the VMIs used: 485 packages (340 MB). However, notice that the UI is typically made available on a 24x7 basis as a scientific gateway to EGI. The ability to dynamically deploy a new UI in a single click out of the existing RADL recipe is very interesting for sysadmins, which can deploy different instances on demand depending, for example, on the number of users that require access to an UI.

*http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
†http://bowtie-bio.sourceforge.net/bowtie2/
‡http://samtools.sourceforge.net
§http://www.broadinstitute.org/gatk/
¶http://galaxyproject.org
‖http://hgdownload.soe.ucsc.edu/goldenPath/galGal4/bigZips/
**http://www.eu-emi.eu/
††http://www.egi.eu/

5. CONCLUSION AND FUTURE WORK

This paper has presented a software platform to easily create scientific on demand virtual computing infrastructures. Firstly, it should be pointed out that creating a high level description or recipe (using RADL) of the desired infrastructure is a much more efficient way to provision infrastructures when compared to the traditional way (provision the computational resources, manual installation of software and configuration, etc.). Secondly, this enables repetition and automation. The same infrastructure can be deployed as many times as required. Moreover, the RADL description can be reused and shared so that the definition of other infrastructures can build upon the recipes of previously defined infrastructures. Thirdly, the ability of the platform to interact with different IaaS Cloud backends is of special importance.

Notice that the same RADL definition is employed to create the infrastructure on an on-premise Cloud or in a public Cloud. This makes it very easy to perform Cloud bursting, where a Cloud provider is employed in case no spare hardware resources are available in the on-premise Cloud of the research institution. Therefore, there is no need for an institution to over-provision hardware resources to cope with sudden workload peaks, where many researchers are simultaneously requesting computing resources to deploy their virtual infrastructures, since the deployment of some virtual infrastructures can be outsourced to a third-party Cloud provider.

Notice that the usage of dynamically deployed infrastructures enables always to start off with a fresh installation. This means that all the users' data and modifications are lost when the infrastructure is torn down. Any unauthorised modification of the infrastructure (even malware, rootkits, etc.) will be purged when the infrastructure's resources are relinquished. This provides a safe, pristine environment every time an infrastructure is deployed.

The usage of such a platform enables research centers to better take advantage of existing hardware resources. By leveraging the elasticity of the Cloud platform, in terms of rapidly provisioning resources, and the advantages of virtualization, such as isolation and multi-tenancy, the proposed platform paves the way for a versatile creation of scientific infrastructures. The functionality of the platform has been tested with three different use cases. In each use case a specific infrastructure with a complex configuration has been deployed.

Since the platform API and tools can be considered stable, future works involve improving the web applications of the platform in order to enhance the user interaction for the less experienced users. This opens avenues to democratize the access to computing resources on demand to support research activities, even for scientists that belong to areas different from computer science.

All the software of the platform is provided to the community as open source code. It can be downloaded from the IM web page* or from the GRyCAP public git repository in GitHub†. Also, a publicly available deployment of the platform is also available to be freely used from the web interface‡.

ACKNOWLEDGEMENTS

The authors would to thank the Spanish Ministerio de Economa y Competitividad for the project Clusters Virtuales Elsticos y Migrables sobre Infraestructuras Cloud Hbridas with reference TIN2013-44390-R.

REFERENCES

1. Mell P, Grance T. The NIST Definition of Cloud Computing. NIST Special Publication 800-145 (Final). *Technical Report* 2011. URL http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.
2. Buyya R, Broberg J, Goscinski AM. *Cloud computing: Principles and Paradigms*. Wiley, 2011. URL http://onlinelibrary.wiley.com/doi/10.1002/9780470940105.fmatter/pdf.

*http://www.grycap.upv.es/im
†http://github.com/grycap
‡http://goo.gl/raSzqq

3. Sahoo J, Mohapatra S, Lath R. Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues. *2010 Second International Conference on Computer and Network Technology*, IEEE, 2010; 222–226, doi:10.1109/ICCNT.2010.49. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5474503.
4. OpenStack. OpenStack 2013. URL http://openstack.org.
5. Sotomayor B, Montero R, Llorente I, Foster I. Capacity leasing in cloud systems using the opennebula engine. *Cloud Computing and Applications* 2008; **2008**:1–5.
6. Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, Zagorodnov D. The Eucalyptus Open-source Cloud-computing System. *Proceedings of 9th IEEE International Symposium on Cluster Computing and the Grid*, 2009.
7. Amazon Web Services. AWS CloudFormation. URL http://aws.amazon.com/cloudformation/.
8. Amazon Web Services. AWS OpsWorks. URL http://aws.amazon.com/opsworks/.
9. Keahey K, Freeman T. Contextualization: Providing One-Click Virtual Clusters. *Fourth IEEE International Conference on eScience*, 2008; 301–308.
10. Keahey K, Freeman T. Architecting a Large-Scale Elastic Environment: Recontextualization and Adaptive Cloud Services for Scientific Computing 2012.
11. Marshall P, Keahey K, Freeman T. Elastic Site: Using Clouds to Elastically Extend Site Resources. *Proceedings of the 2010 IEEE/ACM 10th International Conference on Cluster, Cloud and Grid Computing*, CCGRID '10, IEEE Computer Society: Washington, DC, USA, 2010; 43–52, doi:10.1109/CCGRID.2010.80.
12. Bresnahan J, Freeman T, LaBissoniere D, Keahey K. Managing appliance launches in infrastructure clouds. *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*, TG '11, ACM: New York, NY, USA, 2011; 12:1—-12:7, doi:10.1145/2016741.2016755.
13. Apache. Whirr 2013. URL http://whirr.apache.org/.
14. Juve G, Deelman E. Automating Application Deployment in Infrastructure Clouds. *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science*, CLOUDCOM '11, IEEE Computer Society: Washington, DC, USA, 2011; 658–665, doi:10.1109/CloudCom.2011.102.
15. OASIS. Topology and Orchestration Specification for Cloud Applications Version 1.0 2013. URL http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html.
16. Binz T, Breitenbcher U, Haupt F, Kopp O, Leymann F, Nowak A, Wagner S. OpenTOSCA - A Runtime for TOSCA-Based Cloud Applications. *ICSOC*, *Lecture Notes in Computer Science*, vol. 8274, Springer, 2013; 692–695.
17. Puppet Labs. IT Automation Software for System Administrators 2013. URL http://www.puppetlabs.com/.
18. Opscode. Chef 2013. URL http://www.opscode.com/chef/.
19. DeHaan M. Ansible 2013. URL http://ansible.cc/.
20. Vogels W. Beyond server consolidation. *Queue* Jan 2008; **6**(1):20, doi:10.1145/1348583.1348590. URL http://dl.acm.org/ft_gateway.cfm?id=1348590&type=html.
21. Carrión JV, Moltó G, De Alfonso C, Caballer M, Hernández V. A Generic Catalog and Repository Service for Virtual Machine Images. *2nd International ICST Conference on Cloud Computing (CloudComp 2010)*, 2010.
22. de Alfonso C, Caballer M, Alvarruiz F, Molto G, Hernández V. Infrastructure Deployment Over the Cloud. *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, IEEE, 2011; 517–521, doi:10.1109/CloudCom.2011.77. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6133186.
23. Caballer M, Blanquer I, Molto G, de Alfonso C. Dynamic management of virtual infrastructures. *Journal of Grid Computing* 2014; doi:10.1007/s10723-014-9296-5. URL http://dx.doi.org/10.1007/s10723-014-9296-5.
24. Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters. *Commun. ACM* Jan 2008; **51**(1):107–113, doi:10.1145/1327452.1327492. URL http://doi.acm.org/10.1145/1327452.1327492.
25. Shvachko K, Kuang H, Radia S, Chansler R. The hadoop distributed file system. *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, 2010; 1–10, doi:10.1109/MSST.2010.5496972.
26. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. *Journal of molecular biology* Oct 1990; **215**(3):403–410, doi:10.1006/jmbi.1990.9999. URL http://dx.doi.org/10.1006/jmbi.1990.9999.