Scalable Software Practice Environments Featuring Automatic Provision and Configuration in the Cloud

Germán Moltó, Miguel Caballer

Instituto de Instrumentación para Imagen Molecular (I3M).
Centro mixto CSIC - Universitat Politècnica de València - CIEMAT
Camino de Vera s/n, 46022 Valencia, España
Email: gmolto@dsic.upv.es,micafer1@upv.es

***Abstract*—This paper describes an architecture to deploy scalable Software Practice Environments (SPE) to support the practice lessons that require computer resources that can be remotely accessed. The architecture enables (i) to dynamically and on-demand provision the required computing resources from different IaaS Cloud providers, (ii) to perform the automatic software configuration to satisfy the requirements of the practical lesson, (iii) to suspend and resume the virtual infrastructure in order to cut down costs during a course and (iv) to support different elasticity approaches in order to create scalable virtual infrastructures. The paper describes the proposed architecture and details a case study that involves deploying the virtual infrastructure of an online course on Cloud Computing with Amazon Web Services (AWS) on top of AWS itself. It also describes scalability approaches that can be employed to provide infrastructure access to SPEs for larger audiences, such as those found in MOOCs.**

Keywords: Cloud computing, virtual infrastructures, automated deployment, elasticity

I. Introduction

The students of Computer Science, specially those of Distributed Computing subjects, require access to different computing infrastructures in order to develop the appropriate skills required to efficiently use them. For example, when developing distributed algorithms, the students require access to a set of computers with the appropriate software tools (i.e., compilers, libraries, debuggers, etc.) that allow them to efficiently program those algorithms during the practice lessons. In the context of this paper, a Software Practice Environment (SPE) includes:

- A hardware configuration that satisfies the requirements of the practice lesson. This includes the CPU architecture, the disk size available for students and also special devices (as an example, a practical lesson on programming GPUs requires access to a GPU on which to run the developed codes).
- A software configuration that satisfies the requirements of the practice lesson. This includes the Operating System (OS), the required software, libraries and utilities required for the students to develop the practice lesson. It also includes the user accounts and the configuration of each account.
- Supporting Data. This includes all the data required to perform the practice lesson. For example, the developed algorithms might require certain input data files to perform some benchmarks.

However, deploying and configuring SPEs is far from being a trivial task. Traditionally, many organizations prepare Golden Images that encapsulate the software and data configuration to perform the duties of some (or all of the) subjects. These disk images are then deployed on the PCs of a physical laboratory. However, these approach exhibits many problems. First of all, it hinders extensibility, since including a new application implies modifying the golden image and redeploy it on all the PCs of the laboratory. Second, using a physical laboratory sets an upper bound to the scalability of the computational resources configured to perform the practice lessons (typically no more than two students per PC).

Nowadays, there are two trends that coexist and which enable to surpass the limitations of traditional approaches when it comes to providing a customized software experience for students. On the one hand, Cloud computing is a model that provides network access to a pool of configurable computing resources which can be rapidly provisioned with minimal effort, typically on a pay-per-use basis in the case of public Clouds [1]. On the other hand, the Bring Your Own Device (BYOD) [2] approach enables students to use their own computers and devices in order to access the subject materials. Specially in the case of online courses, where users are not required to attend a physical laboratory, these two trends can be combined in order to offer the users a remote Software Practice Environment (SPE).

This way, the professor can automatically deploy in a Cloud provider (which involves provisioning the virtual infrastructure and configuring it) right before the course starts the required virtual infrastructure that the students require. In the case of using a public Cloud provider, the cost is proportional to the computational and storage resources consumed (mainly hours of Virtual Machine and GBytes of data stored and transferred). Once the course has finished, the infrastructure is relinquished in order to avoid additional costs. In addition, we propose to suspend the deployed infrastructure during unused hours (for example at night) in order to cut down costs. For online courses with different editions through an academic year, this approach is very beneficial, since a new edition of the course simply involves deploying an instance of the virtual infrastructure, which represents a fresh and new install for the new users (without any potentially malicious modifications by the previous students).

The remainder of the paper is structured as follows. First, section II describes related work in this area. Next, section III introduces the main architecture of the proposed system and describes its main components. Later, section IV describes a case study that involves deploying the virtual infrastructure required to support an online course on Cloud Computing with AWS. Next, section V extends the proposed architecture to consider the case of MOOCs, where a scalable virtual infrastructure is required for a large number of students. Finally, section VI summarises the paper and points to future work.

II. Related Work

This paper focuses on the automatic deployment of a virtual infrastructure in a Cloud back-end to support a Software Practice Environment (SPE) that can be remotely accessed by students.

There are different tools that enable to deploy virtual infrastructures in a Cloud. If we focus on open-source tools, Nimbus [3] is project that provides Nimbus Infrastructure, which enables to create Infrastructure as a Service (IaaS) Clouds. It also deals with application contextualization, enabling to deploy virtual clusters on the Cloud. StarCluster [4] also enables to create and manage distributed computing clusters hosted on Amazon EC2. This is also the case of ViteraaS [5], a tool that provides on-demand high performance computing, in the shape of virtual clusters, for research projects, e-learning and teaching within a private Cloud.

All the aforementioned previous works focuses on clusters of PCs deployed on the Cloud. However, our proposed architecture does not focus exclusively on virtual clusters (although it is also able to deploy virtual clusters in a Cloud). In addition, this work extends previous works with two contributions: i) it leverages dynamic configuration without requiring pre-configuration of the Virtual Machine Images, and ii) it focuses on scalability approaches in order to accommodate a large number of users.

There are also commercial tools that automate application deployment and software configuration. For example, rPath, before it was acquired by SAS in November 2012, provided methods to automate the process of packaging, deploying and updating software stacks across physical, virtual and cloud-based environments. Tools such as Kace or BMC Application Automation provide software deployment tools together with automated solutions in order to deploy and manage software throughout an organization.

In our case, we combine application provisioning from a Cloud provider and application deployment and configuration, relying on open-source *DevOps* tools. This enables to have high level recipes to specify the desired infrastructure and enact them on different Clouds.

III. Proposed Architecture

The proposed architecture enables to dynamically provision and configure virtual infrastructures. It features both horizontal and vertical elasticity capabilities to enable scaling the virtual infrastructure, composed of multiple replicated Software Practice Environments (SPEs), to accommodate an increased or reduced number of students. The students can remotely access them using their own devices, typically via SSH, in order to perform a practice lesson.

The proposed architecture builds on some previous developments which, for the sake of completeness, are summarised here:

- The Resource Application Description Language (RADL) [6] is a high level declarative language that includes the hardware, software and configuration requirements of the virtual infrastructure to be deployed. For example, one could describe the requirements for 10 VMs with GNU/Linux Ubuntu 12.04, JDK 1.7+, the installation of the ImageMagick software and 15 user accounts with a set of pre-defined passwords.
- The Infrastructure Manager (IM) [6] is a service-oriented component that takes as input a RADL description of a virtual infrastructure and it provisions the required resources and configures them in order to satisfy the requirements imposed by the RADL description. The IM supports different IaaS Cloud backends, such as OpenNebula [7], OpenStack [8] and Amazon EC2 [9]. As such, it provides a uniform layer to deploy virtual infrastructures on multiple Clouds with the same RADL document. Provisioning resources from multiple Cloud providers is typically known as Sky Computing [10].
- The Configuration Manager. This component is part of the IM and is in charge of performing the deployment and configuration of software, together with the customization of the Virtual Machines (VM). It automates the creation of user accounts, downloading software packages, modifying files, etc. The Configuration Manager currently supports Puppet [11] and Ansible [12]. Both software packages belong to the *DevOps* category and they allow to create recipes in order to automate software deployment and configuration, thus guaranteeing determinism. Puppet uses a pull approach, where agents installed in the VMs of the virtual infrastructure contact a server for instructions on how to configure the VMs. However, Ansible uses a push approach, where configuration is pushed into the VMs from a central server. In our case, we currently rely on Ansible which has proved to be highly scalable.
- The Virtual Machine image Repository & Catalog (VMRC) [13] enables to index and store Virtual Machine Images (VMIs). A VMI is an encapsulation of a virtual hardware configuration together with an Operating System (OS), a set of applications and data. VMs can then be created as instances of a VMI, thus exposing the configuration specified by the VMI. Unlike other catalogs

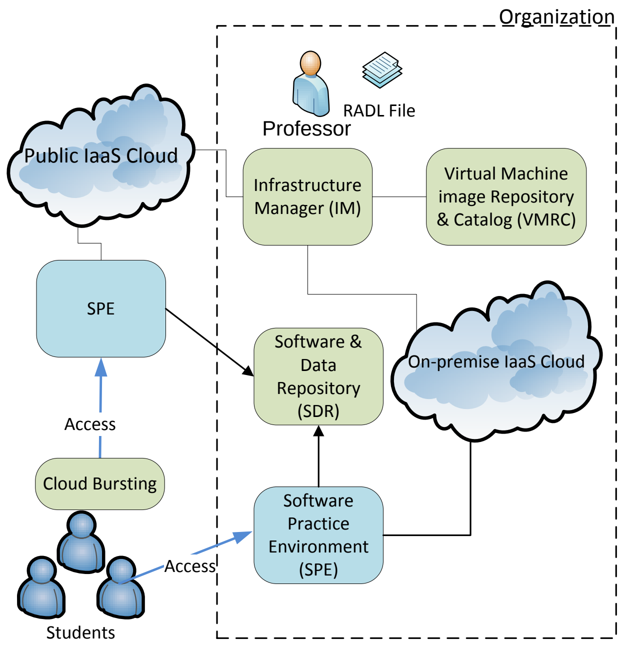
Organization
Professor
RADL File
Public IaaS Cloud
Infrastructure Manager (IM)
Virtual Machine image Repository & Catalog (VMRC)
SPE
Software & Data Repository (SDR)
On-premise IaaS Cloud
Access
Cloud Bursting
Software Practice Environment (SPE)
Access
Students

Fig. 1. Simplified architecture to deploy a multi-node Software Practice Environment in a hybrid Cloud scenario.

of VMIs, VMRC stores metadata of the VMIs (such as the OS, the hypervisor employed to create it, the applications installed, etc.). This metadata can be employed using a query language in order to obtain a ranked list of VMIs that satisfy a given set of requirements (the rank is user-dependent according to the satisfaction of the requirements imposed by the user). For example, one could query the catalog for a suitable list of VMIs based on GNU/Linux Ubuntu greater than 12.04 (hard requirement) and it would be desirable that it had SciLab 4.2+ (this is an example of a soft requirement, where this software can be installed at runtime to satisfy the requirement of the user).

Figure 1 summarises a simplified version of the architecture employed to provision and configure a Software Practice Environment (SPE). The figure assumes an scenario in which the organization (an education center) has an on-premise IaaS Cloud deployment (these are also known as private Clouds, supported by tools such as OpenNebula [7], OpenStack [8] or Eucalyptus [14]). It has also access to a public IaaS Cloud provider such as Amazon Web Services [15], or Rackspace [16], among many others. An IaaS Cloud enables to deploy VMs and manage their life cycles when executing on top of a physical hardware with the help of a hypervisor (or Virtual Machine Monitor) such as KVM [17], Xen [18] or VMware [19].

The idea is to deploy the SPE in the on-premise Cloud and in the public Cloud. This could be performed simultaneously or using a Cloud bursting approach [20], where the public Cloud is only employed when the on-premise Cloud cannot cope with the workload (an increase in the number of SPE instances due to a large number of users). Regardless of the scenario, using a hybrid approach composed of an on-premise and a public Cloud where the same precise configuration exists on both instances of the SPE, introduces fault-tolerance and better ability to workload distribution (different students can connect to different SPE instances).

We summarize the steps required for the professor (or sysadmin) to deploy a SPE using the proposed architecture. First of all, the professor describes the requirements of the infrastructure in a RADL document. This description is submitted to the Infrastructure Manager (IM) which queries the VMRC system to obtain a list of the most appropriate VMIs that satisfy the requirements imposed by the user (the professor in our case). The IM provisions the VMs with the credentials supplied by the professor to access each Cloud infrastructure. Notice that the very same RADL document serves to deploy similar virtual infrastructures with the same configuration. Then, the IM delegates on Ansible to perform all the software installation and configuration. In particular this means:

- To download and install software and required data from both the Software & Data Repository (SDR) or from other external repositories (such as the Ubuntu software repositories). The SDR stores course-dependent data files such as practice guides, input data sets, specific software versions, etc.
- To configure system services. For example, to provide a specific configuration for the SSH server.
- To create user accounts. Using a pre-defined list of account names and passwords, this enables to provide SSH-based access to the SPE for the students.

Once the SPE is up and running, the students connect to it via SSH (or using a graphical desktop via tools such as FreeNX). The ability to define the virtual infrastructure only once in a high-level declarative recipe (using RADL) to deploy a similar virtual infrastructure on different Cloud providers represents a huge step forward when compared to the manual installation and configuration of software. First of all, the professor is now able to perform multiple, deterministic deployments of a similar virtual infrastructure regardless of the Cloud back-end. Secondly, software updates become automatic, since new deployments of the virtual infrastructure, if accompanied by on-demand installation of software, results in an updated virtual infrastructure. Finally, it is possible to deploy new SPE instances on-demand (either in the on-premise Cloud or in the public Cloud) in order to accommodate a larger number of students.

IV. Case Study: The Online Course of Cloud Computing and AWS

The Institute for Molecular Imaging Technologies (I3M) at the Universitat Politcnica de València in Spain offers a three-week online course on Cloud Computing with Amazon Web

Services (AWS)[1]. The course involves theoretical concepts about the Cloud and hands-on practice lessons that demonstrate the usage of the AWS services to create scalable Cloud applications that efficiently access data in the Cloud. The *aws* command-line tool [21] is used to manage AWS services such as Amazon EC2 (Elastic Compute Cloud), Amazon S3 (Simple Storage Service), Amazon SQS (Simple Queue Service) and Amazon SimpleDB. In addition, the official command-line tools to interact with Amazon CloudWatch and Auto Scaling are used. Other services such as Amazon RDS (Relational Database Service) are accessed via a web browser and a database client.

The students connect to a GNU/Linux machine (the SPE) on which they find a pre-configured environment (user accounts, AWS credentials, required tools to interact with AWS). There can be many replicas of this machine since the user state during the practice lessons is always stored in AWS and not in the SPE and, thus, students can connect to whichever instance of the SPE is available. Therefore, students need access to a SPE with the following configuration:

- A VM with GNU/Linux Ubuntu 12.04+, 512+ MB, outbound and inbound connectivity.
- A set of user accounts, each one with the following configuration:
  - A specific username and password pre-allocated by the professor.
  - The Access Key ID and the Secret Access Key to authenticate the student to use the AWS services.
- The following software packages installed:
  - The *aws* tool, described earlier.
  - The Auto Scaling and CloudWatch tools to access those services.
  - A MySQL client (to access databases created with Amazon RDS).
  - OpenJDK JRE 7. This is a requirement for the Auto Scaling and CloudWatch tools.
- The following services configuration:
  - Enable password-based SSH access to the instance (which is disabled by default in Amazon EC2's instances).
- The following data:
  - A package containing the practice guides, sample scripts that demonstrate some AWS services, scripts to populate databases, sample files to be uploaded to Amazon S3, etc.

All this information is specified in an RADL document, which is summarized in Figure 2. The syntax and data has been slightly modified to accommodate the formatting of the paper. Notice that the RADL specifies: i) the physical requirements (such as network with outbound connectivity or a minimum number of RAM), ii) the OS requirements (a minimum version of Ubuntu) and iii) the software and services configuration required in the SPE. Notice that software is automatically

```
network public (outbound = 'yes')
system cursoaws (
cpu.arch='x86_64' and
cpu.count>=1 and
memory.size>=512m and
net_interfaces.count = 1 and
net_interface.0.connection = 'public' and
net_interface.0.dns_name = 'cursoaws' and
disk.0.os.name='linux' and
disk.0.os.flavour='ubuntu' and
disk.0.os.version>='12.04'
)
configure cursoaws (
@begin
- vars:
 - pw_00: O3Je2QxgM0w
 - ak_00: AKIAJAIPMN42O7ADSC5A
 - sk_00: ft0ftS7FD0M5L5Tu3V/
 tasks:
    - user: name=alucloud00 password=$pw_00
    - copy: dest=/home/alucloud00/.awssecret
                content="$ak_00 $sk_00"
 - get_url: url=<sdr_url>/${item} dest=/tmp/${item}
      with_items:
      - cursoaws_1.0_all.deb
      - autoscaling_1.0.61.2_all.deb
      - cloudwatch_1.0.13.4_all.deb
   - command: dpkg -i /tmp/${item}
      with_items:
      - cursoaws_1.0_all.deb
      - autoscaling_1.0.61.2_all.deb
      - cloudwatch_1.0.13.4_all.deb
 - apt: pkg=openjdk-7-jre state=latest
 - get_url: url=<location>/aws
                 dest=/usr/local/bin/aws
  - apt: pkg=mysql-client-5.5 state=installed
  - service: name=ssh state=restarted
@end
)
deploy cursoaws 1
```

Fig. 2. An excerpt of the RADL document to deploy the SPE for the course on Cloud Computing with AWS.

downloaded from the SDR and installed. If files are updated in the SDR, the next deployment of the virtual infrastructure will have updated software. The SSH is automatically configured (not shown in RADL) and restarted to allow password-based connections, which is by default disabled in Amazon EC2.

In this online course, the SPE is deployed on Amazon EC2, although in past courses we also deployed the SPE in an on-premise Cloud based on OpenNebula. Using an on-premise Cloud enables to reduce the costs and offer the same SPE for the students. The number of SPE instances depends on the number of users enrolled in each edition. In order to cut down costs it is convenient to schedule the practice lessons (or at least to have available the infrastructure only during the day, and suspend it at night) in a suspend-resume approach that will be described in the next section.

With the developed system it is possible to deploy, for example, two SPE instances in an average of 7 minutes, involving resource provisioning, software and data downloading and installation and customization (user accounts, ssh configuration, etc.). Notice that multiple instances of SPE are submitted and configured in parallel.

[1]Further information available at http://www.grycap.upv.es/cursocloud

V. Scalable Virtual Infrastructures for MOOCs

With the advent of Massively Open Online Courses (MOOC), we wanted to explore the feasibility of providing a scalable cost-effective access to a Software Practice Environment (SPE) for remote users. Popular courses enroll tens of thousands of students. At the moment, the most common approach to provide a SPE for MOOC students is to prepare a Virtual Machine Image (VMI) with a predefined configuration of the OS, tools and data required to perform the course activities. However, if online services have to be employed (as in the case of the online course on Cloud Computing and AWS), there is no other alternative than providing students with remote access to a GNU/Linux-based SPE so that the student can perform the practice lessons. Some of these courses distribute the cost of using Cloud resources by encouraging students to sign up with a public Cloud provider. In this section, we wanted to explore the possibility that the educational center pays for the infrastructure costs.

Figure 3 describes an architecture to offer scalable SPEs, by leveraging different services from Amazon Web Services (AWS). The proposed architecture is generic enough to be deployed in other public Cloud provider (such as Windows Azure) using the corresponding services.

AWS consists of geographically distributed *regions* across the world which consist of several isolated locations called *availability zones*. The Amazon EC2 service provisions Virtual Machines (called *instances*) from Amazon Machine Images (AMIs). AWS includes many services that fit in the proposed architecture:

- AWS Identity and Access Management (IAM). The professor creates one user credential per student from a single AWS account. These accounts can be temporarily suspended (useful to prevent AWS usage when an edition of the course has finished) and reused by the students of the new edition (since those are not personal accounts).
- Amazon CloudFront. It enables to distribute content at a scale by distributing replicas to different edge locations in the world. Users that request the content will access the nearest replica. This is useful when starting a MOOC with expected peaks in data access (for example, an introductory video accessed by 70k students).
- Auto Scaling. It enables to increase (scale out) and decrease (scale in) the size of the virtual infrastructure. The next subsection focuses specifically on scaling approaches.

A. On Scaling the Virtual Infrastructure

When considering a variable number of users requiring access to a SPE, elasticity, or the ability to increase and decrease the number of instances and the capacities of a SPE, is a key feature. AWS supports different elasticity schemes:

B. Horizontal Elasticity

Within a region, the Auto Scaling service enables to create fleets of instances (an *auto scaling group* (ASG)) that can shrink and grow according to some elasticity rules based mainly on workload or schedule. The ASG includes an Elastic Load Balancer (ELB) that distributes incoming requests for the ELB to the instances of the ASG. The elasticity rules can indicate for example that if the average CPU usage of the instances of the ASG exceeds a 70% during the last 3 periods of 5 minutes, then increase the ASG with 4 additional instances (there are similar rules to scale in).

This approach can accommodate new online students that are performing the practice lessons. If workload is increased, the ASG is increased, and new students that connect to the ELB will be forwarded to an instance of the SPE. Since all the SPE instances are clones and provide the same environment, it does not matter which instance fulfills the request. However, if shared state among the different instances of SPEs is mandatory, these data can be stored in Amazon S3 and pulled by the user to the local instance upon login in the SPE.

Notice that ELB at the moment only supports HTTP(S). Therefore, if access via SSH is required to the SPE then another load balancer such as HAProxy [22], or specific solutions for SSH load balancing such as Ballast [23] should be employed.

In fact Figure 3 depicts an scenario with a two-level load balancing scheme. The students connect to an instance of HAProxy (there could be several of them) which distributes the requests among different ASG in different regions. As such, this provides an scalable approach to perform access to a SPE for multiple students.

Notice that horizontal elasticity automatically manages the number of SPE instances to accommodate an increased or decreased number of students accessing to perform the practice lessons.

C. Vertical Elasticity

In the right lower side of Figure 3, a vertical elasticity approach to scalability is shown. Vertical elasticity is the ability to modify the performance features of a Virtual Machine in order to accommodate an increased (scale up) or reduced (scale down) workload.

Many hypervisors support the ability of dynamically increasing the memory of a running VM without downtime (see for example [24] for a case study with the KVM hypervisor). However, in the case of Amazon EC2, the performance features of an instance cannot be modified without downtime.

Amazon EC2 offers different instance types that range from *m1.small* (1.7 GB of RAM, 160 GB of disk, 32-bit or 64-bit CPU architecture) to *cr1.8xlarge* (244 GB of RAM, 32 virtual CPUs, 240 GB of SSD disk).

AMIs in Amazon EC2 can be of two types: (i) instance-store, where changes in the filesystem are lost when the instance is terminated and (ii) EBS-backed, where an EBS volume (block-based storage) is attached to the instance to store the filesystem changes. An EBS-backed instance can be started (where a per-hour cost for the running instance and a per GB-month for the allocated EBS volume is charged). These instances can be stopped and thus, only the per GB-

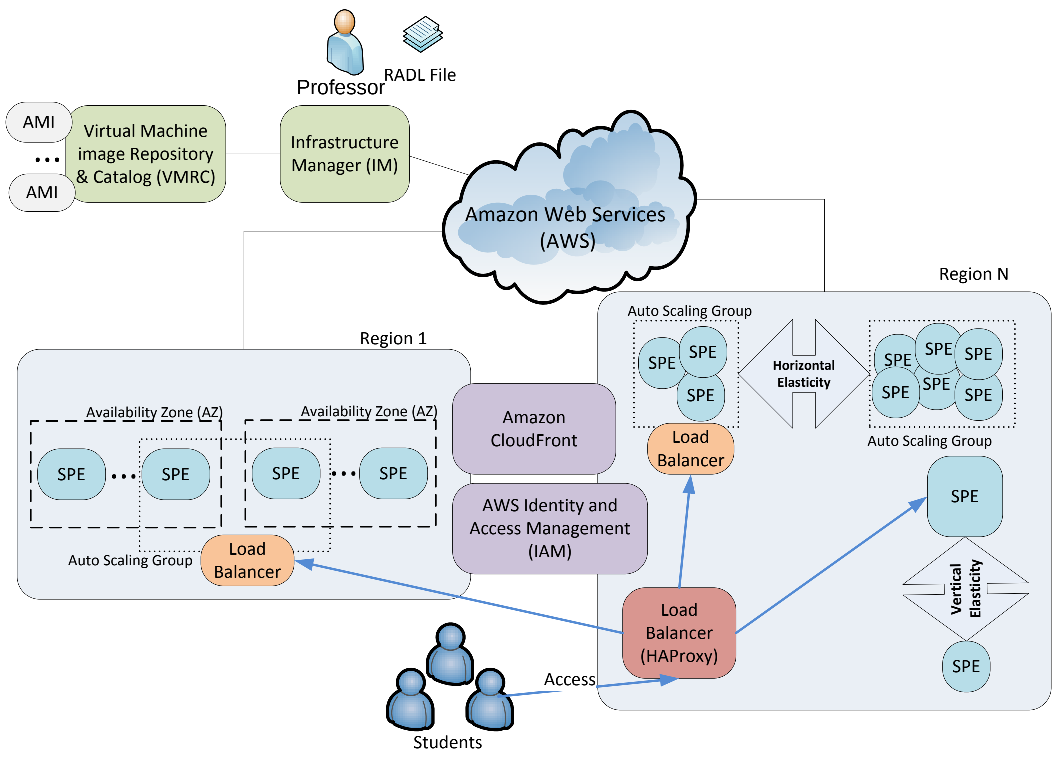

Fig. 3. An architecture to provide scalable Software Practice Environments for MOOCs in Amazon Web Services.

month cost applies (which is in the order of $0.10 per GB-month in the region on Virginia as of June 2013).

Therefore, vertical elasticity can be achieved by stopping the instance, modifying the instance type for an increased or reduced performance and start the instance again, to be able to accommodate a larger workload (a larger number of users). However, the resumed instance changes its IP, a problem that can be circumvented by using Amazon's Elastic IP, an IP address that can be dynamically allocated to different instances (by making the resumed instance to attach itself to the Elastic IP).

D. *Virtual Infrastructure Life Cycle*

Depending on each course, the SPE might be available 24x7 for students to perform the practice lessons at anytime (probably because there are students from different time zones, as in the case of MOOCs). However, consider an scenario in which practice lessons are performed at scheduled intervals (or only during the day). Then it is possible to suspend the virtual infrastructure so that only storage costs (of the EBS volume) are charged. This assumes a suspend-resume approach of the virtual infrastructure, like the one depicted in Figure 4.

When an edition of the course starts, the professor automatically provisions and configures the virtual infrastructure (composed by the SPEs) so that they are ready for users to access them via SSH.

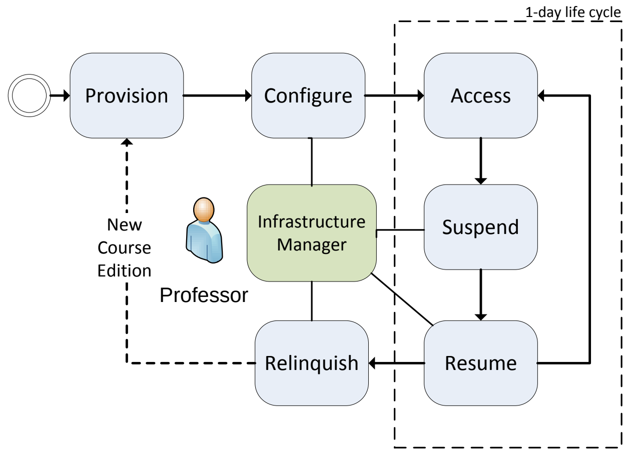

Fig. 4. Flow diagram with the life cycle of a virtual infrastructure.

To have an idea of the costs, consider the following scenario of a 3-week course with 400 enrolled students and 10 SPEs to accommodate 40 students per SPE (a GNU/Linux box to which students connect via SSH to use some online services). Deploying a virtual infrastructure 24x7 with 10 *m1.medium* instances on the Virginia region, and 10 EBS volumes with 80 GB each, costs $955 per month. If you implement a suspend-

and-resume approach to maintain the SPEs only accessible for 8-hours a day the cost can be cut down to \$369.80, thus achieving a reduction of 61%.

Having the infrastructure suspended enables to have the virtual infrastructure ready for service much faster (in the order of a minute) than dynamically deploying and configuring the virtual infrastructure from scratch (which can be performed in the order of 7-10 minutes depending on the complexity of the recipe).

Once the course has finished and the infrastructure of SPEs is no longer required it can be torn down to avoid unnecessary costs.

VI. Conclusions and Future Work

This paper has proposed an architecture to dynamically deploy virtual infrastructures to create Scalable Software Practice Environments (SPE) in IaaS Cloud providers. The infrastructure features automatic provision of computational resources from multiple Cloud back-ends, with the help of the Infrastructure Manager (IM). It also provides automatic deployment and configuration of software and data into the SPEs, with the help of Ansible.

The usage of the architecture has been described to create the SPEs required for an online course. In addition, the architecture has been extended to accommodate larger number of students such as those typically found in popular MOOCs.

The ability to specify in a high level language a declarative description of an infrastructure and to let the system provision, deploy and configure it represents a step forwards towards the widespread adoption of Cloud technologies in online education.

Future works involves providing this tool as a SaaS application so that external users can access its functionality to deploy on other Clouds on behalf of the user. We also plan to extend the tool in order to coordinate the deployment of complex virtual infrastructures (hybrid clusters, Grids, etc.) on the computational resources of an education center. These technologies can greatly simplify the administration of computing resources in an educational center, dealing with the multiple configurations required by the different subjects or courses.

Acknowledgements

The authors would like to thank the financial support received from the Generalitat Valenciana for the project GV/2012/076 and to the Ministerio de Economía y Competitividad for the project CodeCloud (TIN2010-17804)

References

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing. NIST Special Publication 800-145 (Final)," Tech. Rep., 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] M. R. Rafael Ballagas, "BYOD: Bring Your Own Device," in *Proceedings of the Workshop on Ubiquitous Display Environments, Ubicomp*, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.9939

[3] K. Keahey and T. Freeman, "Contextualization: Providing One-Click Virtual Clusters," in *Fourth IEEE International Conference on eScience*, 2008, pp. 301–308.

[4] MIT, "StarCluster." [Online]. Available: http://web.mit.edu/stardev/cluster/

[5] F. Doelitzscher, M. Held, A. Sulistio, and C. Reich, "ViteraaS: Virtual Cluster as a Service," *wolke.hs-furtwangen.de*. [Online]. Available: http://www.wolke.hs-furtwangen.de/assets/downloads/CRL-2010-03.pdf

[6] C. de Alfonso, M. Caballer, F. Alvarruiz, G. Molto, and V. Hernández, "Infrastructure Deployment Over the Cloud," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science*. IEEE, Nov. 2011, pp. 517–521. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6133186

[7] B. Sotomayor, R. Montero, I. Llorente, I. Foster, and F. de Informatica, "Capacity leasing in cloud systems using the opennebula engine," *Cloud Computing and Applications*, vol. 2008, pp. 1–5, 2008.

[8] OpenStack, "OpenStack." [Online]. Available: http://openstack.org

[9] Amazon, "Amazon Elastic Compute Cloud (EC2)." [Online]. Available: http://aws.amazon.com/ec2

[10] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes, "Sky Computing," *IEEE Internet Computing*, vol. 13, no. 5, pp. 43–51, Sep. 2009. [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/MIC.2009.94

[11] P. Labs, "Puppet," http://www.puppetlabs.com, 2010. [Online]. Available: http://www.puppetlabs.com

[12] AnsibleWorks, "Ansible." [Online]. Available: http://ansible.cc

[13] J. V. Carrión, G. Moltó, C. De Alfonso, M. Caballer, and V. Hernández, "A Generic Catalog and Repository Service for Virtual Machine Images," in *2nd International ICST Conference on Cloud Computing (CloudComp 2010)*, 2010.

[14] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-source Cloud-computing System," in *Proceedings of 9th IEEE International Symposium on Cluster Computing and the Grid*, 2009.

[15] Amazon, "Amazon Web Services (AWS)." [Online]. Available: http://aws.amazon.com

[16] Rackspace, "Rackspace." [Online]. Available: http://www.rackspace.com

[17] A. Kivity, Y. Kamay, and D. Laor, "KVM: the Linux virtual machine monitor," *Proceedings of the Linux Symposium*, pp. 225–230, 2007. [Online]. Available: http://www.kernel.org/doc/ols/2007/ols2007v1-pages-225-230.pdf

[18] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*. ACM, 2003, pp. 164–177. [Online]. Available: http://portal.acm.org/citation.cfm?id=945462

[19] VMware, "VMware." [Online]. Available: http://www.vmware.com

[20] S. K. Nair, S. Porwal, T. Dimitrakos, A. J. Ferrer, J. Tordsson, T. Sharif, C. Sheridan, M. Rajarajan, and A. U. Khan, "Towards Secure Cloud Bursting, Brokerage and Aggregation," in *2010 Eighth IEEE European Conference on Web Services*. IEEE, Dec. 2010, pp. 189–196. [Online]. Available: http://dl.acm.org/citation.cfm?id=1932685.1932867

[21] T. Kay, "aws-simple access to Amazon EC2 and S3 and SQS and SDB and ELB." [Online]. Available: http://timkay.com/aws/

[22] HAProxy, "The Reliable, High Performance TCP/HTTP Load Balancer." [Online]. Available: http://haproxy.1wt.eu

[23] NASA, "Ballast." [Online]. Available: http://ti.arc.nasa.gov/opensource/ballast/

[24] G. Moltó, M. Caballer, E. Romero, and C. de Alfonso, "Elastic Memory Management of Virtualized Infrastructures for Applications with Dynamic Memory Requirements," in *International Conference on Computational Science (ICCS 2013)*, 2013.