2013 IEEE Sixth International Conference on Cloud Computing

# Avoiding lock-in: timely reconfiguration of a virtual cloud platform on top of multiple PaaS and IaaS providers

Paulo Rupino da Cunha
*CISUC, Department of Informatics Engineering, University of Coimbra*
*Coimbra, Portugal*
*rupino@dei.uc.pt*

Paulo Melo
*INESC Coimbra and University of Coimbra*
*Coimbra, Portugal*
*pmelo@fe.uc.pt*

Catarina Ferreira da Silva
*CISUC, University of Coimbra and CNRS Lyon Research Center for Images and Intelligent Information Systems*
*Claude Bernard Lyon 1 University*
*Villeurbanne, France*
*catarina.ferreira-da-silva@liris.cnrs.fr*

***Abstract*— We describe our work with a major telecom company in creating a broker that enables them to retain independence from the various PaaS and IaaS providers that they use to support their own SaaS offer on the cloud. To achieve this goal, the broker starts by setting up the required operating environment across the desired mix of PaaS and IaaS providers, and then installs and configures the telco's software on top of that virtual platform. The automation and articulation of these procedures confers the company a considerable flexibility. By updating the list of preferred PaaS providers maintained by our system and forcing a redeploy of the whole environment, it can move services from one supplier to the next, or even to virtual machines running in-house, in a matter of minutes. The most favorable combination of outsourcing and insourcing can be constantly pursued, by pondering factors such as cost, SLAs, and other factors on a provider-by-provider basis.***

***Keywords: Cloud; SaaS; PaaS; IaaS; lock-in; reconfiguration***

## I. INTRODUCTION

Several companies are considering the use of the cloud as a way to reduce their operational expenditures (OPEX) and capital expenditures (CAPEX). However, the concern of becoming locked-in to the service provider is also quite common. This was also the case of our partner company. Besides offering telecommunications services, they also develop and sell the underlying Operations Support Systems (OSS) – the software solutions in charge of provisioning and running the telecommunications services (e.g. voice, Internet, TV). These systems are expensive and their installation and configuration is complex and time-consuming, making them inaccessible to smaller operators. It makes sense, thus, to offer them in the form of Software-as-a-Service (SaaS), by taking advantage of the economies of scale already achieved by third party Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS) providers for required components (such as processing power, or a specific database for storage, for instance). Nevertheless, it is important to retain the flexibility to timely change those providers according to business concerns (such as cost, reliability or performance) or even to keep in-house some of the components that are not available for renting.

Others have acknowledged this issue [1], [2] and first steps have been taken to address it, notably TOSCA [3] which, however, is still under development and will require the providers to offer specific support.

## II. MODELING AND APPROACH

To address the above concerns we designed a Hybrid Provisioning Architecture, implemented in the form of a broker, that enables the company to maintain the most favorable mix of external providers and internal support at any given time. By knowing all the dependencies of the OSS software (e.g. databases, application servers), the broker sets up a virtual platform (supported on the distributed preferred providers) on top of which it deploys the telco's software.

### A. Conceptual architecture

Existing OSS applications make use of several services, usually on local servers, frequently sporting strong coupling. To allow the use of external providers for those services, the coupling must both be made explicit and made generalizable. Our approach describes the coupling requirements and service internal configuration requirements using a service manifesto (an XML document) stating, for each application, which services it requires and eventual dependencies among those services. The manifesto may also contain configuration instructions for each service to fit it to the particular application requirements.

A broker was created to interpret the manifesto, in order to create the required service instances and configure them so that they can be usable by the application (see Figure 1). The manifesto may explicitly describe the service provider to use or this information can be collected from an external system handling company policies (e.g. a business rules engine). The current implementation of the instantiation and configuration is supported by both, provider-specific and provider-and-service-specific drivers, which encapsulate the domain specific knowledge required to support those interactions.

### B. Technical implementation

To demonstrate the architecture, the broker was created and drivers for a few services were developed. Some for

978-0-7695-5028-2/13 $26.00 © 2013 IEEE
DOI 10.1109/CLOUD.2013.36

normal PaaS usage (e.g. to access an Oracle database in the Amazon EC2 cloud, or a Mongo database offered by MongoHQ); others for simulated PaaS providers supported in IaaS providers (e.g. a Zookeeper cluster or a Mongo database instantiated as either fully pre-configured or just bootstrapped virtual machine (VM) images).

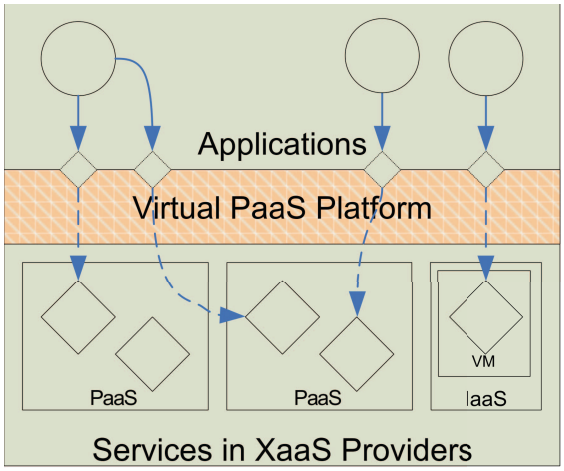

Figure 1 - Conceptual Architecture

A few components of the broker were implemented in a simplified manner. For instance, provider selection resorts to human decisions stored in a database table, but care was taken to support optionally calling external systems capable of suggesting the most suitable choices according to required service functional and non-functional characteristics.

### C. *Validation with OSS application migration to the cloud*

In order to validate the applicability of this approach, we applied the concept to an existing telecommunications OSS application, which, until then, was run on-premises, using in-house supporting software (databases, networking proxies, etc.). We developed each one of the required drivers to support each service on at least one PaaS provider (when available) and on at least one virtual machine supported on an IaaS provider. To support bootstrapping virtual machines rather than fully configured virtual machines, we made use of a systems configuration platform, and therefore were required to instantiate/configure also the platform manager.

### D. *Findings and limitations*

The concept of using a broker to create an on-demand virtual PaaS platform proved successful, in that the existing OSS application could be made to run, without modifications, atop a mix of external PaaS and IaaS providers, and systems running on the internal company servers.

However, some limitations were patent in the approach: the automated provisioning of services can lead to delays that can make the instantiation and configuration of all the providers referenced in section B require 30 minutes from beginning to a fully functional system. This is in spite of the broker parallelizing all possible instantiations, thanks to knowledge in the XML manifesto about the service dependencies. We also noticed that, in the current state of the market, many PaaS providers still expect a manual or at least web-based provisioning, rather than a fully automated REST based interaction with their platforms. This makes it harder to recover from running services the information required to use them on other services. Finally, some providers we used weren't well suited to large numbers of concurrent service instantiations, which required our broker to interleave its requests to the various PaaS and IaaS to reduce the simultaneous load on each. In fact the scalability that cloud services promise is not fully delivered on automatic instantiation of those services.

## III. Conclusion and Future Work

We designed a Hybrid Provisioning Architecture and implemented it in the form of a broker that is capable of setting up a virtual PaaS platform made up of the most suitable mix of public cloud and in-house service providers. The same broker handles the deployment and configuration of existing applications on that environment without need for modifications.

Major contributions of this work are 1) the proposition of an architecture that ensures independency from PaaS and IaaS providers; and 2) the automation of the deployment process, enabling existing applications to transparently rely on a virtual PaaS platform.

As part of our ongoing work, we are working on rollback functionalities which are needed when the process of instantiation and configuration of services fails for whatever reason and the already instantiated resources have to be freed.

## References

[1] F. J. Meng, J. Wang, C. Sun, D. X. Duan, and Y.-M. Chee, "Facilitating Business-Oriented Cloud Transformation Decision with Cloud Transformation Advisor," in *2012 IEEE Fifth International Conference on Cloud Computing*, 2012, pp. 949–950.

[2] C. Ward, N. Aravamudan, K. Bhattacharya, K. Cheng, R. Filepp, R. Kearney, B. Peterson, L. Shwartz, and C. C. Young, "Workload Migration into Clouds - Challenges, Experiences, Opportunities," in *2010 IEEE 3rd International Conference on Cloud Computing*, 2010, pp. 164–171.

[3] OASIS, "TOSCA Topology and Orchestration Specification for Cloud Applications Version 1.0," *OASIS Committee Specification Draft 06 / Public Review Draft 01*, 2012. [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA/v1.0/csprd01/TOSCA-v1.0-csprd01.html.