Towards a Unified Management of Applications on Heterogeneous Clouds

Jose Carrasco
Supervisors: Francisco Durán and Ernesto Pimentel

Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga, Málaga, Spain
josec@lcc.uma.es

Abstract. The diversity in the way cloud providers offer their services, give their SLAs, present their QoS, or support different technologies, makes very difficult the portability and interoperability of cloud applications, and favours the well-known vendor lock-in problem. We propose a model to describe cloud applications and the required resources in an agnostic, and providers- and resources-independent way, in which individual application modules, and entire applications, may be re-deployed using different services without modification. To support this model, and after the proposal of a variety of cross-cloud application management tools by different authors, we propose going one step further in the unification of cloud services with a management approach in which IaaS and PaaS services are integrated into a unified interface. We provide support for deploying applications whose components are distributed on different cloud providers, indistinctly using IaaS and PaaS services.

Keywords: Cloud applications, multi-deployment, TOSCA, Brooklyn

1 Introduction

In recent years, Cloud Computing has experienced a growth in the demand of its services. The Cloud promotes on-demand access to a large number of resources throughout three service models, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [1], which allow cloud providers to offer services for current IT requirements, with scalability and elasticity as the most relevant ones, and allow users to tailor the used resources to their needs.

Vendors such as Google, Amazon, Cloud Foundry, etc., have implemented their solutions to this model by developing their own cloud service layers, with custom APIs that expose their resources. Most of these providers offer a set of similar services as regards functionality, but developed according to their own specifications. E.g., each supplier specifies its own Service Level Agreement (SLA) or Quality of Service (QoS), supports a concrete set of technologies, etc. The proliferation of these solutions has also increased the number of issues to be addressed in cloud computing, mainly related to the diversity of providers and their solutions, giving place to the vendor lock-in problem [2], and hampering the portability and interoperability in the definition and usage of services.

Due to this lack of standardisation, developers are often locked-in to concrete cloud environments, since they have to adapt their developments according to the specifics of the vendors that will be used to run their applications. This heterogeneity affects the entire lifecycle of systems, from design time to release/deployment, which complicates the development of portable applications and the integration of services of different providers to achieve cross-deployments. In this context, migrating components between different platforms seems impossible.

Given the current state of Cloud Computing, it looks reasonable to offer to developers mechanisms to deal with the restrictions to the portability and interoperability of applications. From the developers' point of view, we believe it would be very useful to have an environment in which we could build full detailed application descriptions in an agnostic way, supporting the use of services of different offerings to deploy our applications, and abstracting from the constraints of concrete providers. Furthermore, it would make sense to distribute the different modules of an application over services of different providers. This would allow us to optimise the usage of cloud resources, since we could select, for their deployment, and given the requirements of each of the modules of an application, and requirements of the application itself as a whole, the services with best features for each of the modules of our application. Moreover, we plan to go one step further, and analyse the portability between abstraction levels, initially focusing on IaaS and PaaS.

Once applications have been deployed and are running, using services of specific providers, developers may need to modify the cloud environment where the application is being executed due to many reasons, such as an application updating or different cloud events. For example, the performance of services could be altered, e.g., by a modification in the QoS by the provider, affecting the application performance or its cost. Developers could also modify the cloud resources used by their applications, for example, by adding new cloud services to provide new application features. It may be useful counting with mechanisms supporting the management and reconfiguration of cloud applications.

Additionally, it may be useful to users to have facilities for the migration of application modules between different cloud levels in order to maintain the performance and optimise the resources usage and minimise the cost. For instance, given an increase in the workload of an application, it could be beneficial to migrate some of its modules to PaaS, in order to take advantage of the automatic scalability facilities of this kind of services.

The rest of the paper is structured as follows. Section 2 describes our research challenges. The research plan and the current state of our research are explained in Sections 3 and 4, resp. Section 5 exposes our conclusions and future work.

2 Research Challenges

The main goal of this thesis is to develop an environment that offers an homogeneous management of IaaS and PaaS services, and enables a methodology to describe applications and the required target cloud resources, providing developers with mechanisms to improve the portability and interoperability of applications.

Moreover, it will allow users to choose the cloud resources whose features best adapt to their applications' requirements, with support for the deployment of each of the modules of applications using the PaaS or IaaS services that better fit their needs. In the following, we elaborate on the descriptions of our goals.

Unification of IaaS and PaaS cloud services. We plan to develop a common API that will unify cloud services independently of their abstraction level, for IaaS and PaaS. To achieve this, we will analyse the different service features and restrictions in order to find common patterns and abstract them under a unified interface. This unified level will offer to users a transparent and simple usage of different cloud services, allowing them to focus on the functionality of these services, while the complexity of using and integrating their interfaces is hidden by the unified API. We plan to build this API by homogenising services with different properties in order to build a normalised upper layer. Given the existing diversity, trying to homogenise all the functionalities of each provider will most probably not be possible. To minimise this problem, we will try to maintain these functionalities by using lower layers, with the goal of providing as many services as possible.

Description of applications and cloud services. We believe that the way to address the portability and interoperability issues is by developing an agnostic modeling framework to describe applications and the used cloud (IaaS and PaaS) services and resources. With this framework, users will be able to build full-detailed descriptions of their applications, including all the knowledge about the capabilities, requirements, kinds of services to run the application, etc., regardless of the concrete providers over which the application will be finally deployed. We plan to build on current standards, such as CAMP and TOSCA, in order to propose a standardised, powerful and flexible application-modeling environment.

Integration of the modeling and the unified API. A unified API will offer a homogeneous management of different services. An application model will allow us to detail all the knowledge about an application. Then, we believe that by joining both elements, API and agnostic modeling, we will be able to provide an environment which will allow portable applications to be modeled and deployed using the unified API features in a standardised manner, providing a complete application lifecycle management. Then, any services supported by the unified API will be available for users to deploy modeled applications without requiring any knowledge about the concrete provider interfaces.

Development of a functional prototype. We will develop a functional prototype in which we will experiment with the accomplishments related to the previous goals, and to show its viability and to evaluate its advantages and disadvantages.

Post-deploy management of applications. Although not one of the core goals of the work being described, we will also study the implications of our proposal on the management of applications once they have been deployed and are running. Specifically, we will consider aspects such as the monitoring of cloud applications whose modules are deployed using services of different providers, possibly at different levels, and how SLA policies may be specified (e.g., auto-scaling policies).

Hot reconfiguration of applications. Given agnostic application descriptions, it seems natural to consider the possibility of moving application modules from the services they are deployed on to other ones with better features, or for a better adjustment of the application needs. We may even think on performing such reconfiguration operations at runtime.

3 Research Plan

In this section we structure the work of this thesis on the following phases, detailing the tasks to develop for each of them.

Analysis of the related work

- Exhaustive analysis of the state of the art on homogenisation and cloud management. We will review current practical and theoretical proposals and related standards. We will also analyse their implementation plan.
- Systematic analysis of the features and restrictions of the different cloud offerings in order to determine the key aspects to consider when carrying out the proposed homogenisation. This will be made by defining different deployment use cases involving different service levels.
- Study of deployment-related concepts using services of multiple clouds (multi-clouds).
- Review of related open projects, with special emphasis on those using standards, including an evaluation of their capabilities and limitations.

API composition and unification of IaaS and PaaS services

- Classification of different cloud services in terms of their functionalities and the services of the cloud offerings that will be supported by our approach, establishing a preliminary approach of the unified API.
- A first prototypical development of the unified API. We will most probably first develop independent versions for IaaS and PaaS, which will later be unified under a common interface.
- Our implementation efforts will be integrated inside an existent open project supported by an active community. We will pay special attention to Apache Brooklyn[1], an open project that offers a flexible and robust management of IaaS services of a large number of providers.

Application Modeling

- Analysis of the different concepts related to the management of applications and cloud services that will be supported by our modeling facilities to provide flexible and extensible mechanisms to describe systems.
- Development of a modeling proposal, supporting the definition of applications according to the results of the previous step, addressing the significant management and capabilities differences between the different providers. We will

[1] Apache Brooklyn: `https://brooklyn.apache.org/`.

also study the use of current standards, initiatives and open projects focused on the normalisation of applications and the description cloud services.
- Development of a generic nomenclature to identify and reach the target providers that will be used to deploy applications, making sure that the nomenclature is flexible enough to support any provider properties, and enabling the distribution of the different application modules over different providers (cross-deployment).

Validation of the proposal
- Revise the diversity of use cases proposed on the first phase focusing on different characteristics in order to check the supported providers under diverse restrictions.
- Application of the use cases to specific deployment scenarios which will be composed by different providers according to real situations.

Post-deployment strategies
- As possible extensions, we will consider the monitoring concepts and mechanisms to add them to the common API and the application modeling.
- We will research on management policies, such as auto-scaling, which will be based on the previous monitoring experiments.
- We will study migration techniques, determining how application modules can be moved between services of different providers and its abstraction levels.

4 Current State

We present in this section some of the goals we have already achieved.

4.1 Trans-cloud management

Independent tools and frameworks have emerged with the goal of integrating, under a single interface, the services of multiple public and private providers (see, e.g., [3], and [4]). In a very short time, these platforms have evolved according to the mode in which developers can take advantage of integrated cloud services to expose and run their systems. Terms such as multi-cloud [5], cross-cloud [6], federated clouds [7], or inter-clouds [8] have been used for deployment platforms with the ability of distributing modules of an application using services from different providers.

The main differences between these approaches lie on the different ways of handling the connections between modules deployed on different platforms. However, in all these attempts, platforms allow operating simultaneously with a single level of service to deploy applications, i.e., all the components of an application are deployed either at the IaaS level or all at the PaaS level (see, e.g., [6], [9] and [10]). From this, with the goal of unifying cloud services, we propose in this thesis a second dimension in which deployment tools integrate IaaS and PaaS levels under a single interface. Then, this will allow developers to deploy their applications combining services offered by providers at any of these levels. Following the evolution in terminology, *multi-/cross-/inter-cloud*, we envision *trans-cloud* management tools without the limitations we currently have. *Trans-cloud* mechanisms enable one of most important goals of this work, the unification of IaaS and PaaS cloud services (see Section 1). The idea behind *trans-cloud* is to be able to build our applications by using available services and resources offered by different providers, at IaaS, PaaS or SaaS level, using virtual machines or containers, according to our needs and preferences. We will focus on IaaS and PaaS in this thesis.

4.2 Application Modeling

There is a lot of work on methodology descriptions in the literature, including many projects, standards and initiatives, as Cloud4Soa [10] CAMP[2], Roboconf[3], Terraform[4] and mOSAIC[5] After analysing the most relevant related work, we consider TOSCA (Topology and Orchestration Specification for Cloud Applications)[6] as a standard that provides a useful framework on which basing our application modeling because it defines a very flexible model for the description of cloud applications, the corresponding services, allowing their relations to be specify explicitly by using a fully service topology, containting all the knowledge about the applications. Furthermore, it allows the description of procedures to manage services using orchestration processes by using plans.

Currently, we only take advantage of the topology specification of TOSCA, what allows us to describe the knowledge about applications independently of any cloud resource restrictions, and integrate the different features and requirements of the different provider abstraction levels in the same model.the application portability.

4.3 Toward a Unified API

We propose the development of a common API to unify the management of IaaS and PaaS services. After analysing the mechanisms to manage the cloud of different alternatives, such as OpenTOSCA[7], Alien4Cloud[8], Cloud4Soa and, we decided to base our work on Apache Brooklyn, an open project with an active community behind. Brooklyn can manage the provisioning and deployment of cloud applications, can monitor applications' health and metrics, and handle the dependencies between components. It enables cross-computing features through a unified API to manage IaaS services offered by various providers.

Brooklyn provides an API for the management of IaaS cloud services for a great number of providers and establishes a lifecycle for the management of services and applications. We have extended this API with facilities for the management of PaaS services of platforms based on Cloud Foundry, providing an homogeneous access to IaaS and PaaS services. We have integrated the PaaS

[2] CAMP Standard: `https://www.oasis-open.org/committees/camp/`.
[3] Robocobf: `http://roboconf.net/`.
[4] Terraform: `https://www.terraform.io/`.
[5] mOSAIC: `http://www.mosaic-cloud.eu/`.
[6] TOSCA: `http://docs.oasis-open.org/tosca/TOSCA/v1.0/`.
[7] OpenTosca: `http://www.iaas.uni-stuttgart.de/OpenTOSCA/`.
[8] Alien4Cloud: `http://alien4cloud.org/`.

management in all the Brooklyn levels but without modifying its API. Then, we have obtained a prototype with a common API that manages IaaS and some PaaS services (currently, Cloud Foundry-based platforms) in an unified manner. We have tested this API by building portable applications, and deploying them using different IaaS and PaaS providers. Indeed, we have obtained in this way a first implementation of the proposed trans-cloud mechanisms. However, the current model only supports IaaS and Cloud Foundry-based platforms, so it will be necesary elaborate on PaaS levels of different providers in order to understand their capabilities and requirements and analyze how they would be integrated in our approach by means of extending the current model.

Although we mentioned in Section 2 that post-management is not one of the main goals of this work, but we think the knowledge about this issues would be useful to enhance our IaaS-PaaS integrated model, e.g., migration of applications shares the concern of the management of different providers in order to move the application. Hence, we have developed proofs of concept of some scaling policies both for IaaS and PaaS services by taking advantage of Brooklyn's capabilities.

5 Conclusions and Future Work

We propose the development and use of a common API to unify the management of IaaS and PaaS cloud services, making their use completely uniform. We allocate this proposal inside what we call *trans-clouds*, which extends cross-cloud application deployment and management by supporting the portability and interoperability of application modules from different providers and at different levels. We propose a TOSCA-based agnostic modeling of applications and cloud services, which allows us to specify the characteristics and requirements of any system to be deployed in the cloud. The standardised description of applications and cloud resources and the homogenous service API significantly reduce the portability and interoperability issues related to vendor lock-in, facilitating the reusability of cloud services. By having an agnostic model of our system may greatly simplify migration, or simply decision change. Indeed, with our approach, each component may be deployed at one level or the other just by changing its location. It is worth noting that the proposed thesis project is not an implementation exercise on an existing deployment tool, but an innovative general approach to ease the cloud deployment of applications, enforcing the independence of both cloud providers and cloud models.

We have developed an operational prototype built on the well-established Apache Brooklyn tool in order to test our trans-cloud ideas. Brooklyn provides support for a large number of IaaS providers. Thanks to our efforts in integrating Cloud Foundry into Brooklyn, it now also provides access to PaaS Cloud Foundry-based providers such as Pivotal Web Services or Bluemix.

Part of the research in this thesis was developed in the context of the Seaclouds project [11], and some preliminary results related to the thesis plan described here have already been published in [12,13,14].

Much work remains ahead. We plan to analyse new providers in order to extend the supported PaaS services and technologies. Thus, current model will be extended in order to integrate PaaS levels of new providers, such as Heroku or OpenShift. Due to providers heterogeneity, the new providets has to be carefully analyzed in order to elaborate on how they should be added to our approach. Furthermore, we plan to study the possibility of using the flexibility and scalability mechanisms available for PaaS to develop management policies to react to applications' events.

References

1. Youseff, L., Butrico, M., Silva, D.D.: Toward a unified ontology of cloud computing. In: Grid Computing Environments Workshop (GCE). (2008) 1–10
2. Androcec, D., Vrcek, N., Kungas, P.: Service-level interoperability issues of platform as a service. In: World Congress on Services (SERVICES), IEEE (2015) 349–356
3. Sellami, M., Yangui, S., Mohamed, M., Tata, S.: PaaS-independent provisioning and management of applications in the cloud. In: 6th Intl. Conf. on Cloud Computing (CLOUD), IEEE (2013) 693–700
4. Gonidis, F., Paraskakis, I., Simons, A.J.: A development framework enabling the design of service-based cloud applications. In: Advances in Service-Oriented and Cloud Computing. Springer (2014) 139–152
5. Kritikos, K., Plexousakis, D.: Multi-cloud application design through cloud service composition. In: 8th Intl. Conf. on Cloud Computing (CLOUD), IEEE (2015) 686–693
6. Elkhatib, Y.: Defining cross-cloud systems. ArXiv e-prints (February 2016)
7. Paraiso, F., Haderer, N., Merle, P., Rouvoy, R., Seinturier, L.: A federated multi-cloud PaaS infrastructure. In Chang, R., ed.: 5th Intl. Conf. on Cloud Computing (CLOUD), IEEE (2012) 392–399
8. Grozev, N., Buyya, R.: Inter-cloud architectures and application brokering: taxonomy and survey. Softw., Pract. Exper. **44**(3) (2014) 369–390
9. Hossny, E., Khattab, S., Omara, F., Hassan, H.: A case study for deploying applications on heterogeneous PaaS platforms. In: Intl. Conf. on Cloud Computing and Big Data (CloudCom-Asia), IEEE Computer Society (2013) 246–253
10. Zeginis, D., D'Andria, F., Bocconi, S., Gorronogoitia Cruz, J., Collell Martin, O., Gouvas, P., Ledakis, G., Tarabanis, K.A.: A user-centric multi-PaaS application management solution for hybrid multi-cloud scenarios. Scalable Computing: Practice and Experience **14**(1) (2013)
11. Brogi, A., Carrasco, J., Cubo, J., D'Andria, F., Ibrahim, A., Pimentel, E., Soldani, J.: EU project seaclouds - adaptive management of service-based applications across multiple clouds. In: 4th Intl. Conf. on Cloud Computing and Services Science (CLOSER). (2014) 758–763
12. Carrasco, J., Cubo, J., Pimentel, E., Durán, F.: Multi-deployment over heterogeneous clouds with TOSCA and CAMP. In: 6th Intl. Conf. on Cloud Computing and Services Science (CLOSER). (2016)
13. Carrasco, J., Cubo, J., Pimentel, E.: Towards a flexible deployment of multi-cloud applications based on TOSCA and CAMP. In: Advances in Service-Oriented and Cloud Computing. Springer (2015) 278–286
14. Carrasco, J., Cubo, J., Durán, F., Pimentel, E.: Bidimensional cross-cloud management with TOSCA and Brooklyn. In: Intl. Conf. on Cloud Computing (CLOUD). (2016)