Towards a flexible deployment of multi-cloud applications based on TOSCA and CAMP*

Jose Carrasco, Javier Cubo, and Ernesto Pimentel

Universidad de Málaga, Dept. Lengujes y Ciencias de la Computación, Spain
{josec,cubo,ernesto}@lcc.uma.es

Abstract. Cloud Computing platforms offer diverse services and capabilities with own features. Hence, the provider services could be used by end users to compose a heterogeneous context of multiple cloud platforms in order to deploy their cloud applications made up of a set of modules, according to the best capabilities of the cloud providers. However, this is an ideal scenario, since the cloud platforms are being conducted in an isolated way by presenting many interoperability and portability restrictions, which complicate the integration of diverse provider services to achieve an heterogeneous deployment of multi-cloud applications. In this ongoing work, we present an approach based on model transformation to deploy multi-cloud applications by reusing standardization efforts related to the management and deployment of cloud applications. Specifically, using mechanisms specified by both standards, TOSCA and CAMP, we propose a methodology to describe the topology and distribution of modules of a cloud application and to deploy the interconnected modules over heterogeneous clouds. We illustrate our idea using a running example.

Keywords: Heterogeneous Cloud, Cloud application, Multi-deployment, Model transformation, TOSCA, CAMP

1 Introduction

Cloud Computing is a new paradigm which has become increasingly popular in the last years. It defines a model for enabling convenient and on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned [1]. In this model the providers expose these resources as several services through a cloud platform classified in various levels (IaaS, PaaS, SaaS). They can develop, deploy, and sell cloud applications globally without the need of significant investments in IT infrastructure. The provided services are used by the clients to deploy their applications and systems on concrete provides. Hence, the users could select the services from several providers whose properties and capabilities fit with the requirements of the multi-module cloud application, achieving a flexible deployment context that fully adapts the deployment and execution of the application. However, this is a complex task, since the cloud platforms are being conducted in an isolated way by presenting many interoperability

* Work partially supported by projects TIN2012-35669, funded by Spanish Ministry MINECO, FEDER; P11-TIC-7659 funded by Andalusian Gov; FP7-610531 SeaClouds funded by EU; and Univ. Málaga, Campus Excelencia Int. Andalucía Tech.

and portability restrictions and offering similar resources in a different manner. Each provider defines its own API to the exposed services, its non-functional requirements, QoS, add-ons and so on. As a result, cloud developers are often locked in a specific set of services from a concrete cloud environment. Thus, it is complicated to integrate heterogeneous provider services to achieve a multi-deployment of a cloud application [2]. Currently, several organizations propose different approaches to mitigate these issues through the homogenization and normalization of the cloud application descriptions and management. Specifically, there exist two OASIS standards which pretend to solve some of the problems related with portability, automated deployment, interoperability and management of cloud applications: TOSCA (Topology and Orchestration Specification for Cloud Applications) [3] and CAMP (Cloud Application Management for Platforms) [4]. Both standards specify a particular methodology to describe and wrap the cloud application structure (components and relationships), and how they must be orchestrated (by means of a plan) in a portable way to increase a vendor-neutral ecosystem. Moreover, they describe the mechanisms which must be implemented by the clouds to support standard-based application deployment and management. Nevertheless, the standard efforts do not focus on getting an heterogeneous multi-cloud solution, so distributing a complex application over multiple cloud service providers is still a challenging task [5]. TOSCA and CAMP are emergent standards and they do not have official implementations yet. We can consider available approaches that support a large set of characteristics defined by these standards, *i.e.*, OpenTOSCA Environment [6] (for TOSCA) and Brooklyn [7] (for CAMP).

However, these approaches have some disadvantages. On the one hand, although TOSCA is a good option to represent the application topology and the orchestration, the management of a possible TOSCA-compliant deployment (for example, using OpenTOSCA) would be a complex task, since the topology and the orchestration plan should be modified when some module of the application is migrated to a different target provider. On the other hand, although the CAMP-compliant solutions are not mainly focused on obtaining a multi-deployment, they present an appropriate set of properties to obtain this goal following a unified API which wraps the interface of the cloud providers (for example, Brooklyn which uses jClouds [8]). Nevertheless, CAMP lacks of a topology specification, which is crucial to maintain application model in case of monitoring and reconfiguration actions need to be performed over the distribution of the application. In this ongoing work, we discuss our proposal for combining the advantages of both TOSCA and CAMP specifications and their respective approaches. The main contributions of our idea are: (i) to define a flexible methodology to perform heterogeneous deployment of multi-cloud applications, (ii) to analyse the architectural and technical concepts needed to combine both TOSCA and CAMP specifications, (iii) to address vendor lock-in and portability issues, and (iv) to comply with (and contribute to) major standards for cloud interoperability.

The rest of this paper is structured as follows. Section 2 exposes the motivation and challenges of our approach. In Section 3, we present our proposal based on

the combination of TOSCA- and CAMP-compliant solutions to provide a flexible multi-cloud deployment. In Section 4, we briefly discuss the advantages of our work with respect to current cloud initiatives, and we present some future work.

2 Motivation and Challenges

In this section, we motivate our proposal presenting the challenges to be tackle to deploy cloud applications over heterogenous cloud providers.

2.1 Motivating Example

To illustrate our work, we introduce an example related to an *Online Retailing Application*, composed of four modules: a main Webpage to access the application, two databases (one for the users and another one for the products' stock), and a payment module.

This application could be deployed as a whole on a provider in IaaS or PaaS level, Google(`https://cloud.google.com`), Amazon (`http://aws.amazon.com`), HP Cloud (`http://www.hpcloud.com`), etc. These Cloud providers offer a range of different technologies each appropriate for particular types of applications. So, users can access computing resources in a dynamic, flexible and scalable manner to deploy the mentioned cloud-based application, where each computing resources has its own capabilities, constraints, life cycle and specification (*e.g.* pricing policies, Service Level Agreement (SLAs)). Also, the modules of the application possess own features and requirement. In this sense, it would be interesting to develop a methodology capable of selecting the provider services whose specification fit with the application's requirements and features in order to compose the best heterogeneous deployment context for the distribution of the modules. A large number of companies are trying to simplify the speed and adoption of their products and services to the cloud. The main issue is the lack of interoperability among different vendor approaches, which complicate the deployment over several providers simultaneously.

2.2 Challenges

To perform the multi-deployment, our approach addresses these main challenges:

- **Topology specification.** An application is composed by several modules and relationships, which is essential to maintain the knowledge about the application structure, dependencies among modules and how they are related. We pretend to specify the topology and distribution using a TOSCA-compliant methodology, which allows the maintenance of the application model if some redistribution is required.
- **Unified interface of cloud providers.** Currently, the application developers need to know the interface of the final cloud providers where their applications will be deployed. Our approach proposes to solve this necessity by unifying the features of the heterogeneous platforms by means of a CAMP-compliant approach.
- **Interoperability and portability issues.** In an heterogeneous distribution context, interoperability and portability problems occur. Using our proposal based on the unified interface, the deployment will be in charge of solving

these issues related to the heterogeneity of cloud providers, managing the services needed by the application's modules in an homogeneous manner.
– **Scalability and elasticity resources.** Our deployment process allows the users to consider the scalability and elasticity advantages of cloud provider in order to the best deployment scenario.

3 Proposal in a Nutshell

We present our proposal for the TOSCA- and CAMP-compliant multi-cloud deployment.

3.1 Heterogeneous deployment using TOSCA and CAMP

As shown in Figure 1, our methodology consists of two phases well-defined. Initially, in the first step, we propose to specify the full application topology using TOSCA methodology through the OpenTOSCA environment, specifically the Winery tool (TOSCA-compliant), which allows the description of the applications structure in an exhaustive and user-friendly graphic way. Moreover, we also propose to enrich the TOSCA specification by including information about the final providers where each component of the application will be deployed, with the purpose of facilitating the orchestration according to the expected multi-cloud distribution. In the second step, the application will be distributed over the target clouds through the deployment mechanisms used by Brooklyn (CAMP-compliant). In order to solve the connection between both specification, we propose a transformation methodology to adapt the TOSCA-compliant specification defined to the CAMP-compliant specification expected by Brooklyn.

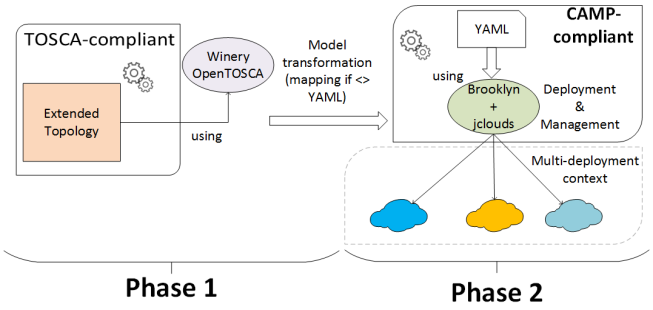

Fig. 1: Overview of our TOSCA- and CAMP-compliant multi-deployment.

3.2 Phase 1: Topology description and orchestration

TOSCA allows the specification of a detailed application topology using an XML-based file. We use TOSCA to obtain a full description of the application modeling its components (application's modules) and dependencies. Then, TOSCA description is composed by several components which present types, properties, requirements, capabilities, and relationships among them. Thus, modifying, deleting or adding some components could provoke an error-prone task due to the need of maintaining the consistency of the initial topology description. In order to solve this problem, we propose to use the Winery tool [9], developed by

the OpenTOSCA Team. This tool allows the representation of the application's modules through forms and the composition of the topology in a graphical way by means of the drag-and-drop technique. In Figure 2 is presented the topology for our running example using Winery.

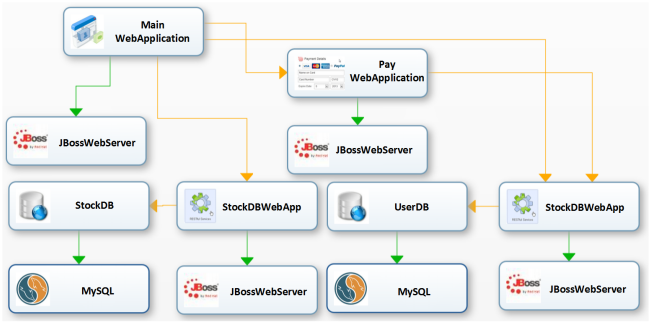

Fig. 2: TOSCA topology for the Online Retailing Application using Winery.

Although we have mentioned the TOSCA expressiveness, this standard does not define in the topology any property to specify the target provider to deploy the application's modules. Instead, it defines an orchestration plan and the implementation artifacts to specify the deployment operations. However, this information does not appear into the topology description, so we need a methodology which allows the clear specification of the final providers. Also, currently, the definition and maintenance of an orchestration plan is a complex and error-prone task.The plans have to fully define each necessary step to deploy and configure the application taking into account all properties and requirements of the providers. Morever, if the target cloud changes, the definition of the mentioned steps must also change in order to reference to the services exposed by the new selected cloud. Therefore, we propose an extension of the TOSCA topology to allow the inclusion of the final providers which are needed to perform the multi-deployment orchestration. In the following text we can see as the NodeTemplate defines a new item to define the **location** where the node (representing the application's module) has to be deployed. To exemplify this extension, the next text models the **UserDBWebApp** by specifying the cloud provider 'AWS' as location.

```
<NodeTemplate id="xs:ID" name="xs:string"? type="xs:QName"
  ...
  <Properties>
          XML fragment
  </Properties>
  <location provider= "xs:string">
  ...
</NodeTemplate>
```

```
<NodeTemplate id="UserDBWebApp" name="User␣Web␣App"
   type="ns3:UserDBWebApplication">
   <Properties>
      <ns3:UserDBWebApplicationProperties
         ...
```

```
      </ns3:UserDBWebApplicationProperties>
    </Properties>
    <location>aws-ec2:us-west-2</location>
</NodeTemplate>
```

Note that the location denition could be modeled like a property in the Node Templates. This could be very usefull to avoid a large negotation of the consortium to approve the standard extension and the providers could feel free to support this feature implementing the necessary mechanisms in their platforms. However, although the mentioned approach has several advantages, our goal goes beyond, by defining an extension of the standard to ensure the performance of the multi-deployment and allow the correct denition of the used providers.

3.3 Phase 2: Transformation model and deployment

The second phase of our proposal is in charge of distributing the application's modules over the different target providers. To tackle these issues, we build this process using a CAMP-compliant environment to take advantage of the homogeneity features aimed by the standard. Then, we use Brooklyn, which (in the latest releases) allows the description of a YAML-based multi-deployment plan. We have defined the topology using TOSCA in the first phase, since CAMP lacks of a topology specification. Therefore, we need to unify the TOSCA-compliant topology definition and the CAMP-compliant deployment mechanism. We propose a model transformation to obtain a Brooklyn YAML plan from TOSCA topology description, by means of two possible transformation processes. The first one is based on an agnostic graph, as depicted in Figure 3. Taking advantage of TOSCA topology definition (similar to a graph structure) we can generate an intermediate graph containing all the details of the application's modules and their relationships. From this agnostic graph we can generate the final (orchestration) plan deployment expected by several technologies, *e.g.,* the CAMP-compliant used in this work, Brooklyn. This task is performed following a set of transformation patterns from the TOSCA-compliant to CAMP-compliant elements (see Table 1). The second proposal is based in meta-model transformations to expose a formal methodology, as shown in Figure 4. In this context, it is necessary to define the meta-model of TOSCA-extended and the Brooklyn plan, together with the ATL rules required to transform a concrete (topology) TOSCA model into a Brooklyn YAML concrete plan.

4 Discussion and Conclusions

In this section, we mention some projects, initiatives and standards in the same scope of our proposal, with the intention of briefly discussing about the contributions of our approach, and finally, we present the future work.

There are several initiatives and standards that target services deployed on the cloud using different approaches, with the consequence that software developers need to either use special APIs or programming models to code their applications, or to model them using project-specific domain languages. The Broker@Cloud project (`http://www.broker-cloud.eu/`) aims at helping enterprises to move to the cloud while enforcing quality control on the developed

Table 1: Transformation pattern between TOSCA and Brooklyn (CAMP)

TOSCA	YAML CAMP
JBoss	brooklyn.entity.webapp.jboss.JBoss7Server
Apache Tomcat	brooklyn.entity.webapp.tomcat.TomcatServer
Jetty Server	brooklyn.entity.webapp.jetty.Jetty6Server
MongoDB Server	brooklyn.entity.nosql.mongodb.MongoDB
Cassandra Data Base	brooklyn.entity.nosql.cassandra.CassandraNode
MySQL Data Base	brooklyn.entity.database.mysql.MySqlNode
Postgre	brooklyn.entity.database.postgresql.PostgreSqlNode
Cluster	brooklyn.entity.webapp.ControlledDynamicWebAppCluster

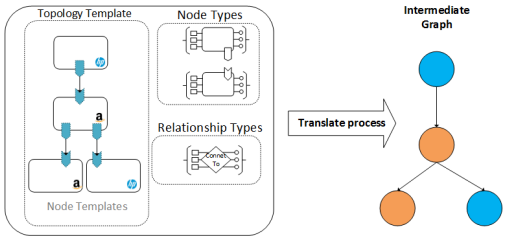

Fig. 3: Abstraction of the TOSCA topology using an agnostic graph.

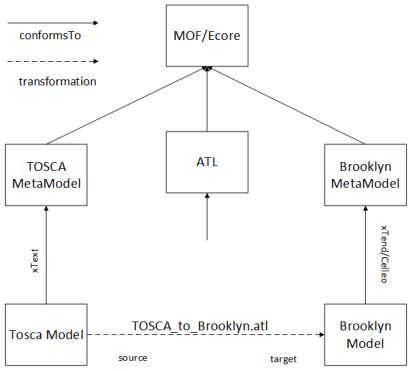

Fig. 4: Meta-model transformation between TOSCA and Brooklyn.

services. The PaaSage project (`http://www.paasage.eu/`) also intends to match application requirements against platform characteristics and make deployment recommendations and dynamic mapping of components to the platform(s) selected. The aim of the Cloud4SOA project (`http://www.cloud4soa.eu/`) is to solve the semantic interoperability issues that exist in current cloud platforms and infrastructures. The mOSAIC project (`http://www.mosaic-cloud.eu/`) allows applications can be deployed on different IaaS using a sort of mOSAIC virtual machine. The REMICS project (`http://www.remics.eu/`) focuses its work on developing advanced model-driven methodology and tools for the reuse and migration of legacy applications to interoperable Cloud services. Cloud standardisation is one of the most active lines in Cloud research. Relevant associations like IEEE or OASIS are working on standards in order to tackle the interoperability and portability between Cloud platforms. The Guide for Cloud Portability and Interoperability Profiles is among the active IEEE projects. TOSCA and CAMP are two OASIS standard concentrating efforts in reducing the deployment and management of cloud applications. These and other Cloud standards, such as DMTF Cloud Infraestructure Management Interface, can be found in the Cloud Standards Wiki (`http://cloud-standards.org/`). In the scope of commercial solutions, we can find some new platforms and open source initiatives that are working on providing flexibility to users allowing the IaaS selection, and in some cases the migration over some specific PaaSes, such as the Cloud Foundry

Core (http://core.cloudfoundry.org/), which defines a baseline of common capabilities to promote Cloud portability across instances of Cloud Foundry.

A distinguish aspect of our approach is that we propose a flexible multi-cloud deployment and management via orchestration and therefore it does not require code modifications to existing services. Thus, we base on the two OASIS standards TOSCA and CAMP to represent the application topology and orchestrate the distribution, and to deploy the modules of the application in multiple and heterogeneous platforms, respectively. Indeed, TOSCA provides a powerful modelling language to describe the structure of an application as a typed topology graph, in a portable and vendor-agnostic way. Also, the use of TOSCA topology templates (and of TOSCA node types) to represent an application topology simplifies its management and fosters the reusability of cloud services. Moreover, CAMP offers a unification in the interfaces of the cloud platforms which allows the management of heterogeneous providers's features in an homogeneous way.

Currently, we are formalising the two proposed model transformation options presented in Section 3: (i) Abstraction of the TOSCA topology using an agnostic graph, and (ii) Meta-model transformation schema between TOSCA-extended and Brooklyn specifications. We also pretend to develop both processes, in order to perform real deployment and management of several complex cloud applications. As regards future work, we plan to analyse the orchestration plan specified in the TOSCA specification (currently there is some research efforts proposing a TOSCA YAML), with the intention of extending our proposal by using this methodology and performing the necessary transformation between the TOSCA YAML and the CAMP YAML, which in principle is out of the objectives of our initial attempt to solve the multi-cloud deployment. Also, some monitoring and reconfiguration mechanisms will be studied in order to address the possible migrations of some application's module when it is needed.

References

1. P. Mell, T. Grance: The NIST definition of cloud computing. NIST, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (2011)
2. D. Petcu, G. Macariu, S. Panica, and C. Craciun: Portable Cloud applications: From theory to practice. Future Generation Computer Systems **29** (2013) 1417–1430
3. OASIS: TOSCA 1.0 (Topology and Orchestration Spec for Cloud Applications), V1.0. http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.pdf (2012)
4. OASIS: CAMP 1.0 (Cloud Application Management for Platforms), V1.0. http://docs.oasis-open.org/camp/camp-spec/v1.1/camp-spec-v1.1.html/ (2012)
5. Leymann, F., Fehling, C., Mietzner, R., Nowak, A., Dustdar, S.: Moving applications to the cloud: an approach based on application model enrichment. International Journal of Cooperative Information Systems **20** (2011) 307–356
6. IAAS: OpenTOSCA. http://www.opentosca.org (2012)
7. CloudSoft: Brooklyn project. http://brooklyncentral.github.io/ (2012)
8. Apache: jClouds Project page. http://jclouds.apache.org/ (2014)
9. Kopp, O., Binz, T., Breitenbücher, U., Leymann, F.: Winery–a modeling tool for tosca-based cloud applications. In: Service-Oriented Computing. Springer (2013) 700–704