Propuesta de metodología de despliegue de aplicaciones en nubes heterogéneas con TOSCA

Jose Carrasco, Javier Cubo, y Ernesto Pimentel

Universidad de Málaga
Departamento de Lenguajes y Ciencias de la Computación, España
{josec,cubo,ernesto}@lcc.uma.es

Resumen. Desplegar y controlar una aplicación compleja sobre un conjunto heterogéneo de proveedores es un problema muy novedoso y complejo al que los clientes de las plataformas de *cloud* se deben de enfrentar. Los proveedores exponen sus servicios de acuerdo a especificaciones independientes incurriendo en una falta de portabilidad e interoperabilidad que converge en la problemática conocida como *vendor lock-in*. Han surgido varias propuestas que aportan soluciones a este ámbito, como el estándar TOSCA que permite describir una aplicación y automatizar su despliegue de forma automática sobre un único proveedor. Extendiendo el estándar mencionado, en este trabajo proponemos una metodología de despliegue y orquestación de los componentes de una aplicación en un entorno *multi-cloud* mediante el uso simultáneo de servicios de diferentes proveedores.
Palabras clave: *Cloud Computing*; Despliegue multi-cloud; Componentes cloud; Orquestación de servicios; TOSCA.

1. Introducción

Cloud Computing [1] o Computación en la Nube ha pasado a ser un componente clave de la Internet, tal y como se conoce y explota actualmente. Este modelo es el resultado de la evolución y asociación de diferentes paradigmas y tecnologías (Virtualization, Client-Server-Model, Peer-to-Peer, Grid-Computing). *Cloud Computing* [2] es un término general para definir un modo de computación que implica un conjunto de componentes (normalmente servidores o bases de datos) conectados e interaccionando entre sí a través de la red, normalmente Internet, y hospedados en el *Cloud* o la Nube. Alojados en estas redes distribuidas se encuentran los diferentes sistemas que ofrecen su funcionalidad a través de servicios clasificados en diferentes niveles (*Infrastructure-as-a-Service*, IaaS; *Platform-as-a-Service*, PaaS; *Software-as-a-Service*, SaaS).

Los usuarios del *Cloud* pueden, por tanto, acceder a los recursos de computación de manera dinámica, flexible y escalable, transformando Internet en un mercado global de colaboración de servicios interconectados a través de la Nube. Por lo tanto, lo ideal sería que los usuarios pudieran desplegar sus desarrollos en cualquier entorno *cloud* ofertado por los diferentes proveedores existentes. No obstante, este es un escenario muy optimista que no se da en la realidad [3], ya que los proveedores han desarrollado sus plataformas sin seguir un estándar o

metodología común por lo que cada sistema necesita usar diferentes mecanismos para albergar desarrollos heterogéneos.

Desde el punto de vista de los desarrolladores esta falta de comunicación entre los proveedores se traduce en una falta de portabilidad de sus aplicaciones, lo cual puede tener consecuencias favorables o desastrosas, según el contexto. Cuándo se comienza el desarrollo de una aplicación se escoge un entorno *cloud* (precedido o no por un estudio más o menos exhaustivo) de manera que se integren las necesidades de la plataforma prácticamente desde el nacimiento de la aplicación, así como los requisitos de la aplicación en sí y los módulos o servicios que la constituyen. Además, esta relación puede ser bidireccional ya que los diferentes proveedores suelen ofertar dentro de sus plataformas diferentes servicios muy útiles para los desarrolladores, los cuales los integran y usan en sus sistemas realizando una unificación con el entorno que los convierte en dependientes del mismo, lo que se conoce como el problema del *vendor lock-in*. De esta manera el desarrollador contrae un bloqueo directo con la plataforma en la que se apoya el proyecto, quedando bloqueado ante futuros cambios por parte del proveedor ya que cuanto más evolucionada esté la aplicación más compleja será la tarea de portarla a otra plataforma.

De acuerdo a la descripción anterior parece complicado imaginar una aplicación compuesta por varios módulos distribuidos a lo largo de varias plataformas e interconectados entre ellos que corresponda al concepto de *multi-deployment*, o como vamos a denominarlo a partir de ahora, despliegue heterogéneo o multidespliegue [4, 5].

TOSCA [6] (*Topology and Orchestration Specification for Cloud Applications*) es un estándar de OASIS que propone ciertos mecanismos para permitir la portabilidad entre diferentes nubes, de forma más o menos transparente al usuario. La intención primordial es estandarizar el despliegue automático entre los diferentes servicios *cloud*, con la idea de hacer más fácil el uso de los mismos. Así, TOSCA permite una descripción interoperable muy expresiva de estos servicios: cómo están relacionados, su comportamiento operacional con independencia del proveedor que cree el servicio y la tecnología de alojamiento usada.

Aunque TOSCA afronta de forma sólida la problemática descrita [7], lo hace parcialmente mediante la distribución de los componentes de una aplicación a través de los servicios de un único *cloud*. En este trabajo presentamos una propuesta de metodología para llevar a cabo multidespliegues en plataformas *Cloud*. Para ello, nos basamos en el estándar mencionado anteriormente, TOSCA, con la intención de aportar nuevas características a su especificación en todos sus niveles, para permitir el despliegue interoperable y portable de las aplicaciones descritas en dicho estándar en entornos de *cloud* heterogéneos. De esta manera, proponemos ampliar la funcionalidad del estándar y su aplicación en distintos casos de uso. En nuestra propuesta, pretendemos definir nuevos mecanismos con el objetivo de gestionar la interacción de servicios o módulos de una aplicación *multi-cloud*, tomando como enfoque el concepto de orquestación de servicios basados en *Cloud*, y aplicando la metodología de composición y orquestación de servicios web [8].

El resto de este artículo está organizado según la siguiente estructura. La Sección 2 presenta una breve introducción del estándar TOSCA. La Sección 3 describe nuestra propuesta de multidespliegue, comparando en la Sección 4 otras alternativas disponibles. Finalmente, presentamos nuestras conclusiones en la Sección 5.

2. Breve introducción a TOSCA

La especificación de TOSCA determina un modelo para definir las aplicaciones o sistemas desarrollados y los servicios requeridos para su funcionamiento. Para ello, especifica una descripción de los elementos que componen una aplicación en los que se definen todas sus características, restricciones, naturaleza y comportamiento, (*Node Types*), tal y como se puede observar en la Fig. 1. Para expresar las relaciones entre los distintos elementos del sistema TOSCA establece un conjunto de relaciones (*Relationship Type*) que permiten determinar las conexiones y dependencias entre componentes (p. ej., conexión de una aplicación web y una base de datos). Estos elementos especifican las posibles entradas y salidas que pueden tener los módulos de la aplicación, a los que modelan, sin concretar sus valores finales. Además, para estructurar los *Types*, TOSCA define una jerarquía de tipos para la extensión y la clasificación de los mismos. En cuanto a su gestión, los *Types* definen operaciones que describen su comportamiento, por ejemplo, un servidor puede definir una operación de inicio para ponerlo en marcha y otra para permitir el despliegue de aplicaciones en su interior. Una vez modelados los elementos del sistema, TOSCA especifica la topología como una metodología para definir de forma exhaustiva la estructurar de una aplicación. Esta topología se compone de nodos y relaciones (ver Fig. 1) que no son más que los componentes concretos que forman la aplicación, los cuales están tipados de acuerdo a los tipos mencionados anteriormente. Todos estos elementos componen la Plantilla de Servicios (*Service Template*) de una aplicación descrita en TOSCA.

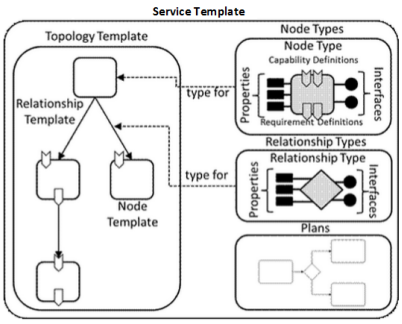

Fig. 1. Proceso de Metaorquestación. (Fuente de la imagen [6]).

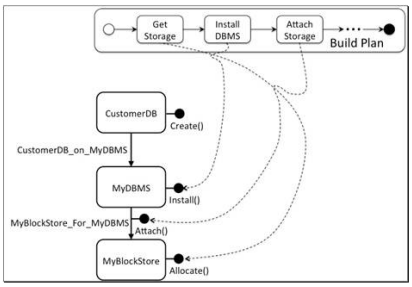

Fig. 2. Metodología de ejecución de planes definida por TOSCA. (Fuente de la imagen [9]).

Dentro de la plantilla de servicio encontramos también los planes de orquestación que especifican mediante un lenguaje de *workflow* (p.ej. BPEL, BPMN, etc.) el proceso de instanciación y gestión de los componentes expresados en la topología. Dichos planes especifican las llamadas de las operaciones de los componentes de la aplicación con el fin de instanciar la estructura descrita en la topología, como se puede observar en la Fig. 2. El modelado de una descripción basada en TOSCA se empaqueta en un archivo contenedor denominado CSAR (*Cloud Service Archive*). TOSCA especifica la estructura del CSAR para que cualquier plataforma que implemente el estándar pueda llevar a cabo la instanciación y orquestación de los componentes que describe, haciéndolo de esta forma portable entre los proveedores.

Para establecer un entorno homogéneo de plataformas *cloud* en el que desplegar los CSAR que modelan las aplicaciones, TOSCA define un conjunto de buenas prácticas en el que se describen los mecanismos necesarios que un proveedor debe de implementar para adaptar el estándar a los servicios ofertados en su plataforma [9], asegurando así la compatibilidad en el proceso de interpretación e instanciación de una aplicación modelada en un CSAR.

Aunque el estándar no cuenta con una implementación oficial, sí podemos encontrar una aproximación que soporta la mayoría de las características de TOSCA, OpenTOSCA [10]. Esta herramienta forma parte de un marco de trabajo que da soporte a todos los aspectos del estándar, definición y construcción de un CSAR [11], así como a la construcción de planes [12]. Sin embargo, esta aproximación plantea varios problemas de de cara la idea propuesta, como veremos en la Sección 4. Otros proveedores públicos (p.ej. Fujitsu, HP, IBM) ofertan sus adaptaciones del estándar [13], los cuales podrían ser utilizados para dar soporte a la idea que se presenta en este trabajo.

3. Propuesta de multidespliegue

En esta sección presentamos nuestra propuesta para realizar despliegues heterogéneos basado en TOSCA.

3.1. Ejemplo de motivación

Para ilustrar la idea presentada, mostramos en la Fig. 3 la distribución heterogénea de los componentes de la aplicación *Online Retailin* que simula un sistema de venta al por menos. En este escenario se despliegan los módulos web en la plataforma de un proveedor (HP) y las bases de datos en otro (Amazon). Además se muestra como se mantienen las conexiones entre módulos, ya estén desplegados en la misma plataforma o no, necesarias para asegurar el funcionamiento correcto del sistema.

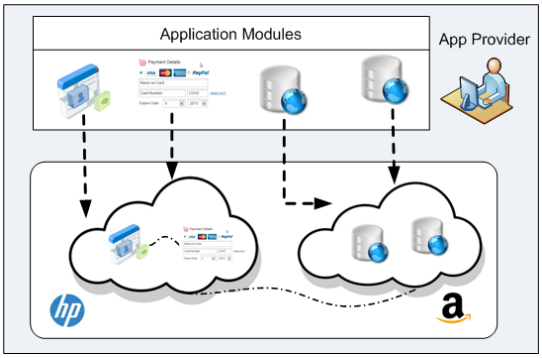

Fig. 3. Distribución heterogénea de la aplicación *Online Retailing*

Esta aplicación podría ser desplegada sobre cualquier proveedor en servicios IaaS o PaaS, por lo que tras la descripción del *vendor lock-in* de la Sección 1 podemos ver como es necesario tener en cuenta las cuestiones de portabilidad. Por otro lado, durante un despliegue heterogéneo hay que conservar la estructura topológica de la aplicación, manteniendo las relaciones y dependencias entre sus componentes, independientemente de los proveedores finales utilizados. Esto se ve afectado por la falta de interoprabilidad entre los proveedores, lo que complica la orquestación entre los servicios necesarios para desplegar los componentes de las aplicaciones y establecer las conexiones necesarias que tienen lugar en el despliegue sobre varios proveedores de forma simultanea.

Con el objetivo de afrontar la problemática de interoperabilidad, portabilidad y uso de contextos *cloud* heterogéneos, descrita en la Sección 1, proponemos una metodología basada en el estándar TOSCA para fomentar el desarrollo de aplicaciones que usen los servicios de múltiples plataformas de forma interoperable.

3.2. Proyección de una aplicación TOSCA

Dado un CSAR, una *proyección* sobre un proveedor *cloud* determina los elementos del mismo que van a ser distribuidos sobre los servicios de su plataforma (es una fragmentación del CSAR) en la que se agrupan todos sus componentes de acuerdo a los proveedores finales cuyos servicios son requeridos.Una proyección, por lo tanto, es una entidad contenedora atómica que alberga todos los componentes de la topología de una aplicación que se van a desplegar sobre un proveedor en concreto, y debe de describir de forma exhaustiva dichos componentes: sus restricciones, requisitos y capacidades, dependencias y relaciones, su gestión y comportamiento (orquestación), etc.

Al recibir un proveedor una proyección, está recibiendo los componentes que deben de ser desplegados en su plataforma junto a toda la información necesaria para llevar a cabo esta tarea. Así, distribuyendo cada una de las proyecciones entre sus plataforma correspondientes se lleva a cabo un proceso de despliegue en un entorno *cloud* heterogéneo donde cada plataforma recibe y contiene exclusivamente los elementos de la topología descritos en su proyección.

Al realizar las proyecciones es necesario mantener la estructura de la aplicación (topología) durante el despliegue de la misma. En este caso, los proveedores finales necesitan algún mecanismo, un plan, donde se les indique los pasos a seguir para realizar el despliegue y la gestión de los componentes que alberga una proyección. Así, la fragmentación de una plantilla de servicio también se realiza sobre su plan de orquestación (obteniendo un plan proyectado) para extraer la información que asegure el despliegue correcto de los componentes de una proyección, manteniendo sus relaciones y dependencias. Gracias a esto, un proveedor es capaz de instanciar los elementos de la topología que le corresponden siguiendo el plan de orquestación adaptado a ellos.

No debemos olvidar que componentes de una aplicación que están relacionados entre sí pueden ser proyectados sobre proveedores diferentes, por lo que es necesario establecer y mantener estas relaciones y dependencias entre plataformas

para asegurar la topología de la aplicación. Esto implica nuevas cuestiones de orquestación, recolección de datos y conexión de componentes que no contempladas en la definición original de TOSCA que pretendemos solventar en nuestra propuesta.

Por estos motivos, proponemos aprovechar la especificación del estándar de TOSCA y definir cada proyección como un CSAR, ya que es un contenedor lo suficiente expresivo como para satisfacer las cuestiones mencionadas. Por supuesto, al igual que sucede en el contexto de TOSCA, los proveedores deben de soportar el estándar (es decir, deben de ser TOSCA-*compliant*) mediante la adaptación de la especificación para poder recibir e interpretar a las distintas proyecciones de una aplicación, tanto los elementos de la topología como los planes proyectados. En este escenario, cada proyección es un CSAR independiente que contiene los elementos de la topología que se van a desplegar en las distintas plataformas, lo que permite:

- **Reutilizar descripciones originales**. En el CSAR que modela una aplicación se describen todos los componentes que forman parte de ella mediante la metodología especificada en el estándar. Usar un CSAR como contenedor de una proyección permite reutilizar toda la descripción de estos componentes facilitando así el proceso de fragmentación.
- **Composición de proyecciones inmediatas**. Dadas las ventajas expuestas en el punto anterior, la composición de las proyecciones será un proceso de agrupación de componentes de acuerdo a los proveedores finales donde van a ser desplegados y a continuación su empaquetado en un CSAR-proyectado siguiendo las reglas impuestas por el estándar.
- **Soporte del estándar TOSCA**. Toda plataforma que haya adaptado sus servicios a la especificación del estándar cuenta con los mecanismos necesarios para interpretar la descripción de los componentes de una proyección. Además, aprovechando los servicios de los proveedores que implementen el estándar necesarios para ejecutar los planes especificados en un CSAR, aseguramos el soporte al de los planes proyectados para desplegar los componentes de la aplicación.

De acuerdo a estos motivos, este escenario establece un entorno de despliegue de aplicaciones en un contexto de *cloud* heterogéneo siguiendo la metodología propuesta en el estándar de TOSCA, tal y como aparece en la Fig. 4. En esta imagen podemos ver el proceso de fragmentación de un CSAR donde se obtiene una proyección para cada una de las plataformas finales utilizadas, AWS y HP Cloud Computing, siguiendo el código de colores naranja y azul respectivamente. Mediante esta nomenclatura se representa la proyección final de los distintos componentes del CSAR, tanto de la topología como de los planes. Cada una de las proyecciones se representa con un CSAR que contiene todos los componentes necesarios, elementos de la topología y fragmentos del plan de orquestación, las cuales son distribuidas entre las distintas plataformas mediante el entorno de adaptación de TOSCA que estas implementan.

Aplicando esta metodología al ejemplo presentado en la Sección 3.1, la aplicación se modelaría mediante un CSAR. Este sería fragmentado según los proveedo-

res utilizados, en este caso se obtendrían dos proyecciones: una para Amazon con las bases de datos y otra para HP con los módulos webs. En este caso, el nivel de los servicios de despliegue utilizados (IaaS o PaaS) va a depender del soporte proporcionados por la implementación del estándar de los proveedores finales.

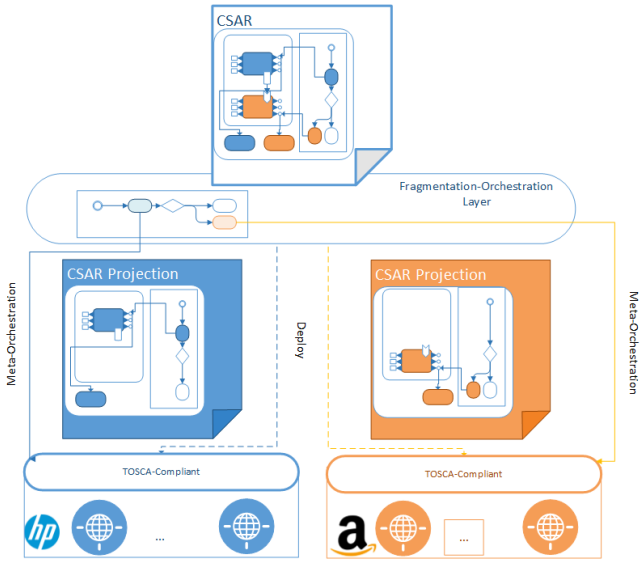

Fig. 4. Proceso de fragmentación, distribución y meta-orquestación de las proyecciones.

En la Fig. 4 observamos una entidad externa a las plataformas *cloud* encargada de la fragmentación de las proyecciones y su distribución a través de los distintos proveedores que denominamos Capa de Fragmentación y Orquestación (*Orchestration-Fragmentation Layer*). El proceso de fragmentación (proyección) de un CSAR no necesita ninguna lógica de adaptación del estándar a ninguna plataforma, no se requiere interpretar el CSAR para la instanciación de la topología sino que solo se va a fragmentar y distribuir las distintas proyecciones a lo largo de las plataformas correspondientes. Esta tarea (fragmentación) es ajena al propio proceso de despliegue de las proyecciones, ya que una vez estas últimas se han distribuido entre los proveedores finales son los entornos de implantación de TOSCA de las plataformas (TOSCA-*compliant*) los encargados de gestionar y desplegar los componentes descritos en dichas proyecciones, obteniendo así un multidespliegue. Consideramos este motivo una base suficiente de la autonomía e independencia de la entidad encargada de esta tarea que, como hemos descrito, es ajena a cualquier plataforma. De esta forma, *(i)* se mantiene el control y la gestión de los componentes desplegados en las distintas plataformas por la capa de adaptación de TOSCA en los proveedores, *(ii)* se mantiene su descripción original que es encapsulada dentro del contenedor especificado por el estándar, y *(iii)* se favorece a la implantación del estándar en nuevas plataformas para

ofrecer sus servicios en despliegues heterogéneos ya que esta capa externa es la encargada de implementar la lógica extra planteada en nuestra propuesta, .

Afrontando estas cuestiones, la capa de fragmentación se encarga de realizar la orquestación entre proveedores, para lo que define un plan de orden superior (*meta-plan*) de orquestación donde se especifica la lectura de estos datos dinámicos y la composición de las relaciones entre los componentes distribuidos. Este meta-plan debe de ser inferido de forma automática del plan original y la topología de la aplicación. Es necesario analizar las relaciones y dependencias entre componentes, comprobando las proyecciones de los mismos para, a continuación, realizar una clasificación exhaustiva de las instrucciones descritas en el plan de orquestación original que operen sobre las relaciones entre componentes con distintas proyecciones. De esta forma, una vez se ha recopilado esta información, se pasa al proceso composición de plan de meta-orquestación extrayendo del plan original los fragmentos necesarios para conectar componentes entre plataformas. Así mismo, podría ser necesario añadirle nuevas instrucciones para completar los posibles huecos que puedan aparecer al conectar secciones del plan original que anteriormente no estaban relacionadas. Con el objetivo de realizar esta tarea, la capa de fragmentación y orquestación requiere *(i)* acceso a los servicios de despliegue de los CSAR en las plataformas finales utilizadas. *(ii)* Estas plataformas deben de ofrecer mecanismos suficientes para realizar la lectura de los datos necesarios y para establecer las conexiones entre los componentes.

3.3. Extensión del estándar

La especificación actual de TOSCA determina un amplio conjunto de características que describen a los componentes de una aplicación basada en el estándar. Sin embargo, a priori, el estándar no facilita ninguna metodología para indicar de forma explícita los proveedores finales donde estos van a ser desplegados, es decir, su proyección. Esto hace imposible determinar el proveedor objetivo de cada uno de los componentes de la aplicación. Es cierto dichos componentes determinan los proveedores donde van a ser desplegados mediante los artefactos de implementación que describen las operaciones y los servicios necesarios en el despliegue y gestión de los componentes de la topología. Aunque en última instancia se podrían usar estos elementos como fuentes de datos para realizar la proyección de un CSAR sobre una plataforma, parece complicado extraer este conocimiento de estos artefactos que pueden ser tan heterogéneos y complejos como el programador que los desarrolla lo desee, por lo que no consideramos que sea una alternativa plausible a las carencias de la especificación en este contexto. Así mismo, estos artefactos no expresan, bajo ninguna circunstancia, la orquestación de los servicios necesarios para obtener un despliegue correcto (información que como hemos visto en el epígrafe anterior es de vital importancia para la composición de las proyecciones). Para hacer frente a esta problemática y facilitar la proyección de un CSAR sobre una plataforma, proponemos una ampliación del estándar con el objetivo de permitir una descripción exhaustiva de los proveedores finales donde van desplegar los módulos de una aplicación.

En la topología de TOSCA los componentes de una aplicación se modelan mediante la plantilla de nodos y relaciones (*Node and Relationship Template*),

ampliando la especificación de estos elementos se pueden definir nuevas propiedades para describir los proveedores finales donde van a ser proyectados. En este sentido, en el Listado 1.1 mostramos nuestra aportación a la especificación de TOSCA[1], donde hemos añadido una cláusula, *projection*, a la definición de los nodos plantilla mencionados para describir las plataformas finales donde van a ser desplegados.

Listado 1.1. Aportación a la definción de TOSCA para especificar los proveedores finales de despliegue.

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

En el Listado 1.1 podemos ver cómo los valores de las proyecciones son simples cadenas de texto. Con el objetivo de expresar de forma inequívoca las proyecciones de los artefactos es necesario añadir a la especificación identificadores estandarizados que hagan referencia a las plataformas de *cloud* que se van a utilizar, lo que facilitaría el proceso realizado por la capa de fragmentación. Por ejemplo un mapa del tipo clave-valor, como el que se muestra en el Listado 1.2, en el que se establezcan los identificadores de las distintas plataformas. En el ejemplo mostrado en el listado aparecen identificadores para referencia a la plataforma de AWS y HP (usadas en el ejemplo descrito en la Sección 3.1) que, en este caso, se especifican hasta los centros de datos seleccionados en el despliegue. Esta metodología ha sido heredada y adaptada de las tecnologías Brooklyn y Jclouds, las cuales veremos en la Sección 4

Listado 1.2. Ejemplo mapa Clave-Valor para identificar a los proveedores.

```
tosca.location.named.AWS\ Oregon\ = aws-ec2:us-west-2
tosca.location.named.HP\ Cloud\ Arizona-1 = hpcloud-compute
```

En la Sección 2 ya se mencionó que el estándar define un conjunto de buenas prácticas para facilitar su proceso de implantación en una plataforma; sin embargo no se especifican mecanismos de acceso a los datos del despliegue de la topología los cuales, como hemos visto, son necesarios para que la capa de fragmentación pueda llevar a cabo la meta-orquestación de las proyecciones estableciendo las relaciones entre componentes distribuidos en distintos proveedores. Por estos motivos, proponemos ampliar la especificación del estándar definiendo una metodología que permita recolectar datos dinámicos de despliegue y establecer relaciones entre componentes. Dicha metodología podría estar compuesta por un conjunto de servicios REST que especifique cómo una plataforma debe exponer estas funcionalidades desde los mecanismos de implementación del estándar, tal y como

[1] La definición original de estos componentes en está disponible en [6]

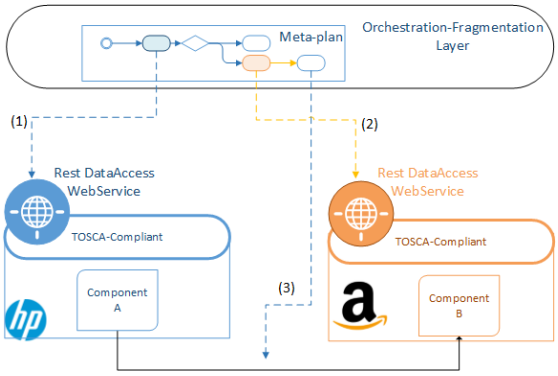

Fig. 5. Plantilla de servicio, topología y demás elementos definidos por TOSCA.

se muestra en la Fig. 5. En esta ejemplifica un proceso de meta-orquestación para conectar dos elementos de una aplicación desplegados en proveedores distintos. Este proceso se compone de tres pasos gestionados por la capa de fragmentación donde se muestra la necesidad de estos mecanismos de comunicación expuestos por las plataformas.

A través de estos servicios la capa de fragmentación se encarga de las siguientes acciones: *(i)* Acceso a los datos del componente A. *(ii)* Envío de datos al componente B. *(iii)* Establecimiento y gestión de la relación entre A y B.

Tras la propuesta que que recoge nuestra idea para llevara acabo despliegues heterogéneos sobre entornos TOSCA-*compliant*, presentamos a continuación el trabajo relacionado.

4. Trabajos relacionados

Existen otras alternativas que ofrecen metodologías para afrontar el desarrollo y el despliegue de aplicaciones en contextos heterogéneos.

La propia organización de OASIS, que ha desarrollado la especificación de TOSCA, propone otro estándar para el despliegue, gestión y monitorización de aplicaciones *cloud* independientemente de la plataforma: CAMP [14] (*Cloud Application Management for Platforms*). En él se define una API común que abstrae a todos los proveedores bajo una misma interfaz, de forma que permite el desarrollo y modelado de aplicaciones apoyadas en esta API, portables a un entorno de plataformas homogéneo. Al igual que TOSCA, CAMP cuenta con una aproximación de implementación estable, Brooklyn (`http://brooklyn.incubator.apache.org/`), un *framework* de control basado en políticas de código abierto para aplicaciones distribuidas que presenta una continua alineación con el estándar, ofreciendo mecanismos para definir aplicaciones de acuerdo a la especificación del estándar, usando la librería (`http://jclouds.apache.org/`) como API. Aunque no deja de ser una aproximación, Brooklyn permite el despliegue de aplicaciones en entornos heterogéneos compatible con todos los proveedores adaptados en Jclouds. No obstante, a diferencia de nuestra propuesta que describe la topología de un sistema, esta herramienta no mantiene una

estructura explícita y expresiva de la aplicación, por lo que una vez realizada la distribución de los elementos es complicado comprobar la estructura de los componentes (cuestión muy importante para controlar el estado la aplicación).

Ubuntu juju (https://juju.ubuntu.com/) también es un *framework* de orquestación que soporta el despliegue heterogéneos de servicios facilitando su interrelación y la descripción topológica, de una forma similar a la de Winery. Sin embargo, aunque está muy extendido en el ámbito comercial, a diferencia de nuestra propuesta no está basado en ningún estándar, lo que desde nuestro punto de vista es esencial de cara a mitigar el *vendor lock-in*.

En la Sección 2 mencionamos OpenTOSCA como una posible aproximación a TOSCA en el que se especifica que la entidad de adaptación de TOSCA debe estar integrada en las plataformas finales, permitiendo así el uso directo de los servicios de los proveedores para el despliegue de las aplicaciones que mantienen un conocimiento exhaustivo de la aplicación; enfoque que mantenemos en nuestra propuesta delegando y confiando el despliegue de las proyecciones en la entidad implementación del estándar por parte de los proveedores. No obstante, OpenTOSCA aunque es un entorno de alta utilidad y usabilidad, es externa a cualquier plataforma. Esta independencia, aunque la consideramos una gran aportación en cuanto a la problemática del *vendor lock-in*, requiere mecanismos externos para el acceso a los servicios de los distintos proveedores (APIs de comunicación), implicando de esta forma una tarea extra de mantenimiento de dichos mecanismos. Por otro lado, a diferencia de nuestra propuesta en la que el conocimiento de los elementos distribuidos en una plataforma se mantenía en la capa de TOSCA-*Compliant*, OpenTOSCA no presenta como objetivo distribuir este conocimiento entre los proveedores finales, ni considera de forma explícita la posibilidad de multi-despliegue en diversos proveedores.

Podemos encontrar otros grupos de investigación y desarrollo que ofrecen sus propias alternativas para solventar los problemas de portabilidad e interoperabilidad y control de plataformas de *cloud*. Por ejemplo, DMTF (*Distributed Management Task Force*) (`http://www.dmtf.org/`) propone *Interoperable Cloud* [15] donde definen varias normas y estándares con el objetivo de definir interfaces entre los clientes y los proveedores de *cloud*, estableciendo así un conjunto de mecanismos que facilitan la portabilidad e interoperabilidad entre las plataformas y los clientes. CIMI (*Cloud Infraestructure Management Interface*) [16] otra posible alternativa desarrollada por CMWG (`http://www.dmtf.org/standards/cmwg`), un subgrupo de DMTF, define la gestión e interacción entre *clouds* a nivel de IaaS estableciendo bajo una API común a todos los proveedores, de la misma forma que pretende hacerlo CAMP. Durante los últimos años tambien han ido ganando importancia los proveedores clasificados como *Cloud Federation*. En términos generales, este tipo de plataformas permiten seleccionar al usuario final los infraestructura sobre la que se quiere realizar un despliegue mediante un conjunto de servicios de terceros. Por ejemplo, Paraiso [17] propone definir un nivel de PaaS sobre el nivel de IaaS de un proveedor.

Aunque estos estándares afrontan el *vendor lock-in*, consideramos que TOSCA cuenta con ciertas ventaja frente a estas especificaciones. TOSCA define un

modelado de los sistemas que van a ser desplegados, que en la mayoría de los casos están formados por un conjunto de componentes heterogéneos interconectados entre ellos. Además, especifica una descripción interoperable de los servicios utilizados mediante un plan de orquestación. De esta forma, no pretende afrontar los distintos problemas de la plataforma mediante la definición e imposición de una API a los distintos proveedores, sino que por el contrario usando este plan define un sistema capaz de usar los servicios que estos exponen. Por otro lado, TOSCA propone una especificación para la topología de una aplicación, describiendo así la estructura de la misma, los elementos que la componen, y cómo se relacionan entre ellos y los distintos servicios de la plataforma. Las ventajas que exhibe TOSCA frente a estos otros enfoques son heredadas por nuestra propuesta.

En este ámbito hemos de mencionar el proyecto SeaClouds (*SEAmless adaptive multi-CLOUD management of service based applicationS* [18]) donde se pretende abordar y apoyar la implementación y administración de aplicaciones complejas multicomponente sobre conjuntos de *clouds* heterogéneos de manera adaptativa. De esta manera se adapta el entorno o contexto de ejecución de la aplicación a las necesidades de la misma, haciendo uso de los estándares y tecnologías que se acaban de describir. Precisamente, la presente propuesta tiene lugar dentro del ciclo de vida del proyecto SeaClouds para abordar cuestiones relacionadas con el despliegue heterogéneo, además de estudiar la alineación con los estándares existentes, previamente mencionados.

5. Conclusiones y trabajo futuro

En este trabajo, hemos propuesto una metodología para realizar el despliegue de aplicaciones en entornos *multi-cloud* tomando como base el estándar de TOSCA, que pretende resolver la problemática de interoperabilidad y portabilidad relacionada con el uso de los servicios de las distintas plataformas de *Cloud Comptuing*.

Nuestra idea sigue la filosofía planteada por el estándar donde las aplicaciones se describen mediante la especificación de: tipos (*Types*) y plantillas (*Templates*) para construir la topología, planes de orquestación para la gestión y control de los mismos y artefactos con los elementos reales que componen el sistema. Una aplicación se distribuirá a través de los distintos proveedores mediante el concepto de *proyección*, que hemos definido como archivos contenedores CSAR, manteniéndonos dentro de la especificación del estándar, y donde serán las capas de adaptación de las plataformas (TOSCA-*compliant*) las encargadas de gestionar el despliegue de todos los componentes proyectadas. Esto permite que, tras un previo estudio, los usuarios puedan optimizar los recursos disponibles diversificando los distintos componentes de una aplicación atendiendo a las características de los servicios ofertados por los distintos proveedores

Esto respeta las aportaciones de TOSCA en cuanto a portabilidad e interoperabilidad entre las plataformas así como todos los esfuerzos realizados para definir una metodología de orquestación de los componentes. La portabilidad y la automatización de despliegue de las aplicaciones que TOSCA describe son cuestiones relevantes que pretenden solventar la problemática del *vendor lock-in*,

características sobre las que hemos pretendido construir nuestra propuesta; dotando así a los usuarios de un entorno de despliegue heterogéneo reutilizable y flexible. Estas cuestiones abren nuevas líneas de investigación sobre aspectos del propio estándar, que constituyen el trabajo futuro que pretendemos abordar:

- **Proyecciones de topología y plan**. Proponemos soluciones para definir la proyección (localización) de los módulos que componen una aplicación de forma inequívoca mediante la adicción de nuevas características a la especificación y la estandarización de los identificadores de las plataformas. Aunque esto posibilita la fragmentación de una topología y los demás componentes, se establecen nuevas líneas de investigación en la descomposición de los planes para orquestar los elementos que componen una proyección. TOSCA describe estos planes como procesos de *workflow*, cuya especificación queda fuera del ámbito del estándar, por lo tanto será necesario un proceso de *matching* entre el plan y la topología de una aplicación para determinar la forma correcta de orquestar los componentes de las proyecciones.
- **Construcción de las proyecciones**. Las proyecciones se encapsulan en CSARs, los cuales ya están especificados en el estándar. Tomando este trabajo como base, el estándar debe de añadir nuevos mecanismos para definir CSAR proyectados sobre las plataformas con el objetivo de optimizar la meta-orquestación de los mismos.
- **Meta-orquestación**. Esta nueva línea de investigación pretende aportar soluciones a las cuestiones relacionadas tanto con el establecimiento de relaciones *inter-clouds* como con los mecanismos necesarios para llevarlas a cabo, lo que implica la ampliación de las capas TOSCA-*compliant* de las plataformas finales.
- **Reconfiguración y variabilidad adaptativa**. Estudio de la posibilidad de abordar la personalización del despliegue heterogéneo utilizando técnicas de variabilidad del software, de forma que los proveedores finales utilizados dependan de un conjunto de restricciones y características identificadas y valoradas por los usuarios finales del sistema.

6. Agradecimientos

Este trabajo ha sido desarrollado con el apoyo de los proyectos: TIN2012-35669, financiado por el Ministerio Español de Economía y Competitividad y FEDER; FP7-610531-SeaClouds, financiado por la Unión Europea; P11-TIC-7659, financiado por la Junta de Andalucía; y la Universidad de Málaga y el Campus de Excelencia Internacional Andalucía Tech.

Referencias

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Communications of the ACM **53** (2010) 50–58
2. Mell, P., Grance, T.: The nist definition of cloud computing (draft). NIST special publication **800** (2011) 7

3. Petcu, D.: Portability and interoperability between clouds: challenges and case study. In: Towards a Service-Based Internet. Springer (2011) 62–74
4. Nicolae, B., Cappello, F., Antoniu, G.: Optimizing multi-deployment on clouds by means of self-adaptive prefetching. In: Euro-Par 2011 Parallel Processing. Springer (2011) 503–513
5. Yang, X., Zhang, H.: Cloud computing and soa convergence research. In: Computational Intelligence and Design (ISCID), 2012 Fifth International Symposium on. Volume 1., IEEE (2012) 330–335
6. OASIS: TOSCA 1.0 (Topology and Orchestration Specification for Cloud Applications), Version 1.0. http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.pdf (2012)
7. Lipton, P.: Escaping vendor lock-in with tosca, an emerging cloud standard for portability. CA Labs Research (2012) 49
8. Camara, J., Martin, J.A., Salaun, G., Cubo, J., Ouederni, M., Canal, C., Pimentel, E.: Itaca: An integrated toolbox for the automatic composition and adaptation of web services. In: Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on, IEEE (2009) 627–630
9. OASIS: TOSCA 1.0 (Topology and Orchestration Specification for Cloud Applications), Primer version 1.0. http://docs.oasis-open.org/tosca/tosca-primer/v1.0/tosca-primer-v1.0.pdf (2013)
10. Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., Wagner, S.: Opentosca–a runtime for tosca-based cloud applications. In: Service-Oriented Computing. Springer (2013) 692–695
11. 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
12. Kopp, O., Binz, T., Breitenbücher, U., Leymann, F.: Bpmn4tosca: A domain-specific language to model management plans for composite applications. In: Business Process Model and Notation. Springer (2012) 38–52
13. OASIS: Interoperability Demo of OASIS TOSCA. https://www.oasis-open.org/events/cloud/2013/toscademo (2013)
14. OASIS: CAMP 1.0 (Cloud Application Management for Platforms), Version 1.0. http://docs.oasis-open.org/camp/camp-spec/v1.1/camp-spec-v1.1.html/ (2012)
15. DMTF: Interoperable Cloud. http://www.dmtf.org/sites/default/files/standards/documents/DSP-IS0101_1.0.0.pdf (2013)
16. CMWG: CIMI. http://dmtf.org/sites/default/files/standards/documents/DSP0263_1.0.1.pdf (2013)
17. Paraiso, F., Haderer, N., Merle, P., Rouvoy, R., Seinturier, L.: A federated multi-cloud paas infrastructure. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, IEEE (2012) 392–399
18. Brogi, A., Ibrahim, A., Soldani, J., Carrasco, J., Cubo, J., Pimentel, E., D'Andria, F.: Seaclouds: a european project on seamless management of multi-cloud applications. ACM SIGSOFT Software Engineering Notes **39** (2014) 1–4