Propuesta de trabajo Final de la Carrera Ingeniería de Sistemas

Aether, un framework para facilitar la implementación de aplicaciones basadas en Cloud Computing

Dirección:

Dr. Alvaro Soria.

Alumnos:

Pérez Fuentes, Joaquín Alejandro,
Steimbach, Marcos Maximiliano.

1. Motivación

El mercado IT[1] (Tecnología informática) está siendo transformado por el modelo de Cloud Computing y la tecnología como servicio. Los expertos creen que en pocos años la gran mayoría de las empresas estarán utilizando la Cloud, de igual modo que adoptaron el correo electrónico o la PC en su momento. Las principales ventajas de Cloud Computing radican en la rapidez y facilidad de implantación y gestión, la flexibilidad, el ahorro de costes, prestaciones técnicas, capacidad de escalabilidad, de forma que estas puedan adaptarse instantáneamente a las necesidades específicas que una compañía pueda tener en un momento determinado, ante un pico de trabajo, sin depender de adquisiciones de servidores o de licencias de software. Esto genera una mayor agilidad en la toma de decisiones empresariales y, consecuentemente, una mayor competitividad de las empresas[2].

Cloud Computing es un modelo de acceso bajo demanda a recursos computacionales compartidos (servidores, almacenamiento, servicios, etc.)[3]. Estos recursos pueden ser tomados y liberados rápidamente con un mínimo esfuerzo de configuración e interacción con el proveedor. Este modelo presenta algunas características principales como son por ejemplo: proveer **self-service de recursos bajo demanda** con lo cual el cliente puede obtener y dejar recursos de manera unilateral sin intervención del proveedor, proveer un **pool de recursos** para poder distribuir los mismos entre múltiples clientes de manera dinámica bajo la demanda de cada uno, permitir un **crecimiento/decrecimiento rápido** ya que el cliente puede obtener y dejar recursos rápidamente acorde a su demanda y proveer además un **servicio medido** lo cual permite a los sistemas en Cloud controlar el uso de los recursos de manera transparente al cliente y facturarle en base a esto (por ejemplo, para máquinas virtuales el uso suele medirse en horas mientras que en almacenamiento se mide en GBs. Todos los proveedores manejan variaciones de este mismo esquema).

Los servicios de Cloud Computing se agrupan en tres tipos básicos:

- SaaS (Software as a Service). Se le provee al cliente la posibilidad de utilizar una aplicación del proveedor que corre en una infraestructura de tipo Cloud. Ejemplos de esto son los servicios de webmail.
- PaaS (Platform as a Service). Se le provee al cliente la habilidad de desplegar una aplicación propia creada con el lenguaje, bibliotecas y herramientas soportadas por el proveedor. El cliente debe adaptarse al ambiente de ejecución provisto. Ejemplos de este tipo se servicio son Google App Engine[4] o Heroku[5].

- IaaS (Infrastructure as a Service). Se le provee al cliente la habilidad de obtener recursos computacionales en los cuales puede desplegar sus aplicaciones arbitrariamente. En este tipo de servicios el cliente no maneja la infraestructura subyacente pero tiene control sobre sistemas operativos, sistemas de archivos, máquinas virtuales y las aplicaciones que despliega. A diferencia de lo que sucede en PaaS, no se le obliga al cliente a trabajar con un ambiente de ejecución impuesto por el proveedor. Ejemplos de este tipo de servicio son Amazon EC2[6], Google Storage[7] o Windows Azure[8].

Existe en la actualidad una variada oferta de proveedores para todo tipo de servicios siendo los más importantes Amazon, Microsoft, Rackspace[9], Google, IBM[10], Hewlett-Packard[11] y VMWare[12]entre otros. Cabe destacar que pese a la diversidad de los servicios ofrecidos, en casi todos los casos el modelo de cobro es por el uso real de los recursos. Existen algunas variaciones a este modelo como por ejemplo la reserva de instancias por año (similar a un servidor usual), paquetes de almacenamiento fijo o descuentos por grandes cantidades de recursos utilizados[13].

Esta gran variedad de proveedores requiere que los desarrolladores de aplicaciones diseñen pensando en esta potencial migración para aprovechar las ventajas competitivas de otras plataformas [14][15][16][17]. Esto es útil ya que múltiples razones pueden llevar a que un usuario desee cambiar de proveedor, incluso teniendo su aplicación en producción. Por ejemplo, los costos de un proveedor pueden bajar y convertirlo en una alternativa interesante. Otro caso podría ser la velocidad de red que posea determinado proveedor para una ubicación geográfica específica. Incluso se podrían usar varios servicios en simultáneo para generar redundancia.
Debido a esto surgen herramientas que tratan de darle solución a este problema siguiendo el enfoque de intentar homogeneizar interfaces. Estas herramientas abstraen una interfaz común que luego adaptan a las interfaces de los proveedores que soportan. Sin embargo, trasladan el problema a la herramienta, dejando dependiente a la aplicación de la herramienta y de los proveedores que esta soporta. jClouds[18], Dasein[19] y libcloud[20] entre otros integran este grupo de herramientas. La limitación principal de estas herramientas yace en que ninguna considera la posibilidad de migrar una aplicación ya codificada con otra herramienta. Supongamos el caso de tener una aplicación codificada con el SDK de Amazon[21]. En un momento dado los creadores de esta aplicación determinan que les resultaría conveniente migrar a Google Storage, pero este no es soportado por el SDK de Amazon. Como consecuencia, el equipo de

desarrollo se encontraría con la situación de tener que recodificar su aplicación con otra herramienta acorde al nuevo proveedor.

2. Propuesta

De acuerdo a la problemática planteada, el objetivo de este trabajo es crear un framework que reduzca el esfuerzo de codificación para migrar una aplicación entre proveedores de servicios de Cloud Computing. La figura 1 presenta a grandes rasgos el funcionamiento del enfoque propuesto.

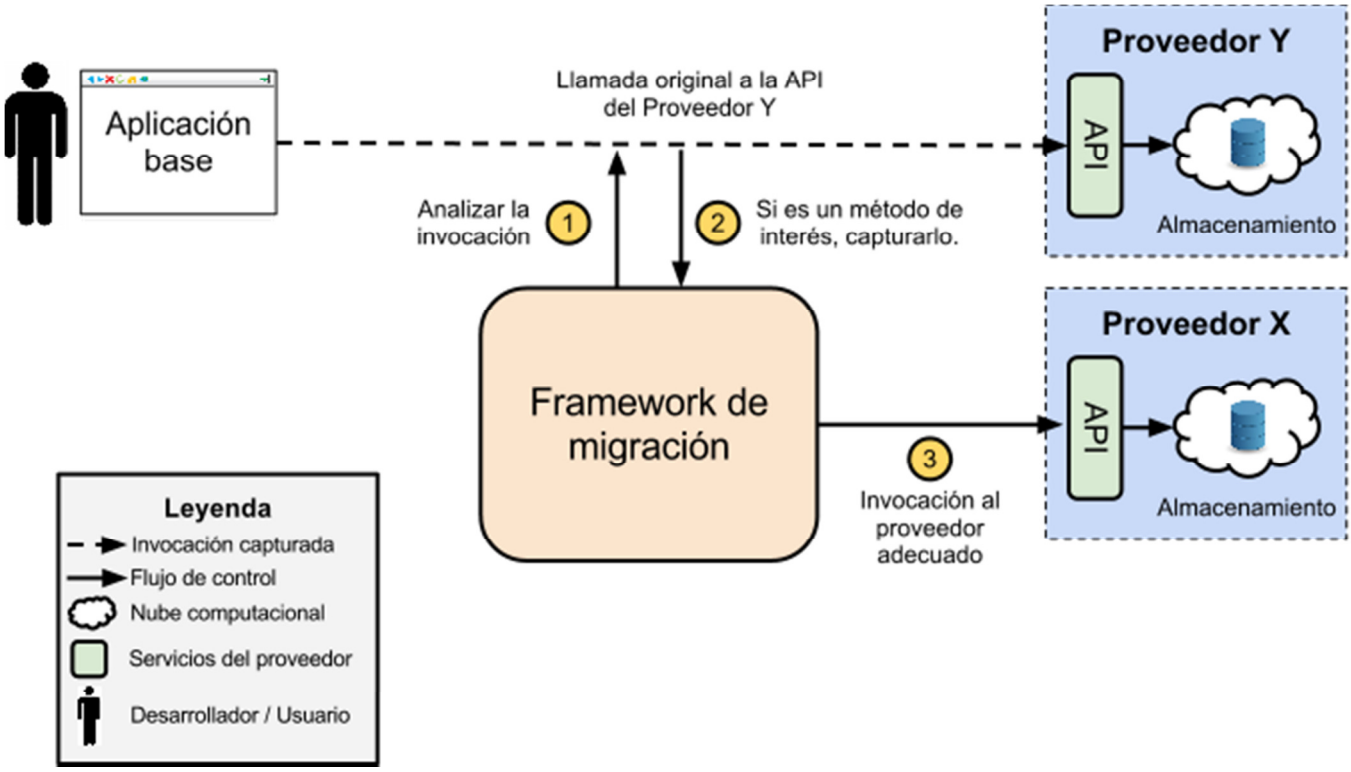

Figura 1, esquema básico del framework de migración.

Como puede apreciarse en la figura anterior, el framework se encuentra analizando cada una de las llamadas que la aplicación del usuario (Aplicación base) realiza a la API del proveedor del servicio Cloud (1). En caso de que una de las invocaciones corresponda a alguno de los métodos buscados se procede a capturarla (2). Para estos dos puntos se explorará el uso de mecanismos basados en reflexión [22] los cuales permiten observar y modificar la estructura de un programa en tiempo de ejecución. Posteriormente, las llamadas capturadas se traducen en llamadas equivalentes al proveedor indicado por el usuario (3). Para éste último punto se analizará la utilización de adaptadores capaces de traducir una

llamada original en un conjunto equivalente de invocaciones para el nuevo proveedor. El usuario del framework deberá encargarse de proveer la información requerida para realizar las traducciones. Entre estos elementos se encuentra el servicio al que se desea migrar, sus credenciales y cualquier otra información que el framework necesite para realizar la traducción.

Cabe destacar que este mecanismo puede aplicarse a cualquier tipo de servicio en Cloud, ya sea almacenamiento, colas distribuidas, procesamiento u otro. Este trabajo se llevará a cabo sobre servicios de almacenamiento y podrá extenderse para incluir soporte para el resto de los tipos de servicios.

3. Cronograma de actividades

A continuación se presenta el cronograma de actividades propuestas -y su duración estimada en semanas- que conformarán el desarrollo de éste trabajo final. El tiempo total de trabajo se fijó en aproximadamente 24 semanas.

Actividad	Duración estimada
Relevamiento bibliográfico	4 semanas*
Análisis de los frameworks existentes para almacenamiento en Cloud	4 semanas*
Planteo de los requerimientos del problema	3 semanas
Análisis y diseño de los requerimientos planteados	5 semanas
Implementación	7 semanas*
Medición y evaluación de los resultados	3 semanas
Documentación y elaboración del informe	8 semanas*

Las actividades marcadas con “*” pueden trabajarse de forma paralela.

4. Referencias

[1] Diccionario de informática, http://www.alegsa.com.ar/Dic/tecnologias%20de%20la%20informacion.php, accedida el día 25/03/2013

[2] El cloud multiplica la competitividad de las empresas, http://cloud.ticbeat.com/cloud-multiplica-competitividad-empresas-entrevista/, accedida el día 12/04/2013

[3] The NIST Definition of Cloud Computing, Peter Mell and Timothy Grance, NIST Special Publication 800-145 (September 2011).NationalInstitute of Standards and Technology.

[4] Google App Engine, https://developers.google.com/appengine/, accedida el día 05/03/13.

[5] Heroku Cloud ApplicationPlatform, http://www.heroku.com/, accedida el día 05/03/13.

[6] Amazon EC2, http://aws.amazon.com/es/ec2/, accedida el día 05/03/13.

[7] Google Cloud Storage, https://developers.google.com/storage/, accedida el día 05/03/13.

[8] Windows Azure, http://www.windowsazure.com/es-es/, accedida el día 05/03/13.

[9] Rackspace, http://www.rackspace.com/, accedida el día 05/03/13.

[10] IBM Cloud Computing, http://www.ibm.com/cloud-computing/us/en/, accedida el día 05/03/13.

[11] Hewlett-Packard Cloud Services, https://www.hpcloud.com/, accedida el día 05/03/13.

[12] VMWare Cloud Computing Solutions, http://www.vmware.com/solutions/cloud-computing/index.html, accedida el día 05/03/13.

[13] Amazon EC2 Pricing, http://aws.amazon.com/es/ec2/pricing/, accedida el día 05/03/13.

[14] Leavitt, Neal. "Is cloud computing really ready for prime time." Growth 27, no. 5 (2009).

[15] Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee et al. "A view of Cloud Computing." Communications of the ACM 53, no. 4 (2010): 50-58.

[16] Hofmann, Paul, and Dan Woods. "Cloud Computing: The limits of public Clouds for business applications." Internet Computing, IEEE 14, no. 6 (2010): 90-93.

[17] Briscoe, Gerard, and AlexandrosMarinos. "Digital ecosystems in the Clouds: towards community Cloud Computing."In Digital Ecosystems and Technologies, 2009.DEST'09. 3rd IEEE International Conference on, pp. 103-108. IEEE, 2009.

[18] jClouds, http://www.jclouds.org/, accedida el día 05/03/13.

[19] Dasein Cloud API, http://dasein-cloud.sourceforge.net/, accedida el día 05/03/13.

[20] Apache libCloud, http://libcloud.apache.org/, accedida el día 05/03/13.

[21] Amazon AWS SDK para Java, http://aws.amazon.com/sdkforjava/, accedida el día 05/03/13.

[22] Ira R. Forman,Nate Forman. “Java Reflection in Action” (2005).