Configuración guiada por búsqueda de aplicaciones basadas en microservicios en la nube

José Antonio Parejo[1*], Aurora Ramírez[2], José Raúl Romero[2], Sergio Segura[1] y Antonio Ruiz-Cortés[1]

Dpto. de Lenguajes y Sistemas Informáticos, Univ. de Sevilla[1],
Dpto. de Informática y Análisis Numérico, Universidad de Córdoba[2]
japarejo@us.es,{aramirez,jrromero}@uco.es,{sergiosegura,aruiz}@us.es

Resumen Organizaciones como Netflix, Google o Amazon hacen uso de arquitecturas basadas en microservicios, lo que ha disparado el interés de la comunidad en ingeniería del software por este estilo arquitectónico. No obstante, el buen uso de este estilo arquitectónico supone nuevos retos, como determinar qué instancias de servicios se despliegan o establecer la mejor configuración de la nube que los aloja, conforme a la carga de trabajo esperada para cumplir los Acuerdos de Nivel de Servicio. Se trata de un problema de optimización en el que deben considerarse simultáneamente múltiples propiedades, a menudo en conflicto entre sí. Por ello, tras formular este caso como un problema de búsqueda, se discutirá cómo el uso de técnicas multi-objetivo puede mejorar las soluciones actuales, permitiéndonos escoger los proveedores y configuraciones apropiadas para disminuir los costes de explotación, y asegurar la disponibilidad de los servicios críticos sin empobrecer el tiempo de respuesta.

Keywords: microservicios, optimización multi-objetivo, *cloud computing*, Acuerdos de Nivel de Servicio

1. Introducción

La evolución de las arquitecturas distribuidas ha desembocado en la definición de un estilo arquitectónico denominado arquitectura basada en microservicios. Este estilo arquitectónico ha sido aplicado con éxito en organizaciones como Google, Ebay [5], Netflix [4], o Amazon [2] para aplicaciones con cargas de peticiones masivas que deben ofrecer tiempos de respuesta bajos, y alta disponibilidad.

En este tipo de aplicaciones los distintos módulos o historias de usuario de la aplicación se implementan como servicios web RESTful independientes. De esta forma, se alcanza un nivel de modularidad que facilita el control del redespliegue en únicamente aquellas partes que soportan mayor carga de trabajo y, consecuentemente, evitan el uso indiscriminado de la infraestructura.

[*] Este trabajo ha sido sufragado parcialmente por la Comisión Europea(FEDER), y los gobiernos de España y Andalucía mediante los proyectos BELI (TIN2015-70560-R), THEOS (TIC-5906), COPAS (TIC-1867) y TIN2014-55252-P, la Red de Excelencia SEBASENet (TIN2015-71841-REDT) y el programa FPU (FPU13/01466).

Sin embargo, diversos autores han señalado que, junto con las ventajas que ofrece este estilo arquitectónico, aparecen también nuevos retos y decisiones a tomar. Una de ellas es determinar la manera en que los microservicios se distribuirán en las distintas máquinas virtuales (MVs) en el caso de realizar un despliegue en la nube. Concretamente, con el objetivo de aprovechar la elasticidad y el modelo de pago por uso que se emplea normalmente en la nube se hace necesario decidir: i) las MVs que se van a usar y sus características concretas, ii) cuántas instancias de cada microservicio se van a desplegar, y en qué maquinas virtuales concretas lo harán, y iii) las reglas de escalado a aplicar sobre la infraestructura y los microservicios para adaptarse a la carga de trabajo conforme a los Acuerdos de Nivel de Servicio (ANS) establecidos. En este artículo, se presenta y formaliza este problema, identificando un conjunto de objetivos de interés que lo configuran como un problema multi-objetivo susceptible de ser abordado mediante técnicas de búsqueda.

2. Ejemplo motivador

Supongamos una aplicación de gestión de personal donde se controla el horario, acceso a las instalacciones, y las tareas que realizan los empleados de una organización durante su jornada de trabajo. Dichas tareas corresponden además a diversos proyectos que la organización desea gestionar. Podríamos identificar 4 microservicios: gestión de usuarios, gestión de instalaciones y control de acceso, gestión de tareas y proyectos, y gestión de sesiones de trabajo (que asocian a usuarios con tareas concretas durante un número de horas). Cada microservicio podría tener asociado una base de datos (BD) para la gestión de su información, que es compartida por todas sus instancias, y se tendrá adicionalmente un registro de instancias de servicios disponibles que es a su vez otro microservicio. El esquema de despliegue más simple posible, es aquel en el que todos los microservicios y bases de datos se despliegan sobre la misma MV, y no se realiza ningún tipo de escalado en base a la utilización. Uno de los esquemas de despliegue alternativo posibles es aquel en el que hay 2 MVs, y cada microservicio tiene un único despliegue en alguna de las MVs. La figura 1 muestra ambos esquemas como diagramas de componentes UML decorados.

3. Configuración de aplicaciones basadas en microservicios en la nube

Sea $S = \{s_1, \ldots, s_n\}$ el conjunto de microservicios a desplegar en la nube. Sea $N = \{n_1, \ldots, n_{MaxMV}\}$ un conjunto de máquinas virtuales, cada una con una configuración de proveedor de cloud determinada. Sea $C = \{c_1, \ldots, c_l\}$ el conjunto de configuraciones posibles de máquinas virtuales proporcionadas por un conjunto de proveedores de infraestructura en la nube. Una configuración de la arquitectura cloud viene determinada por el valor de las variables: $x_{i,j}$, $y_{i,j}$, y z_j, ST y GT. Donde $x_{i,j}$ son variables binarias que indican si el microservicio

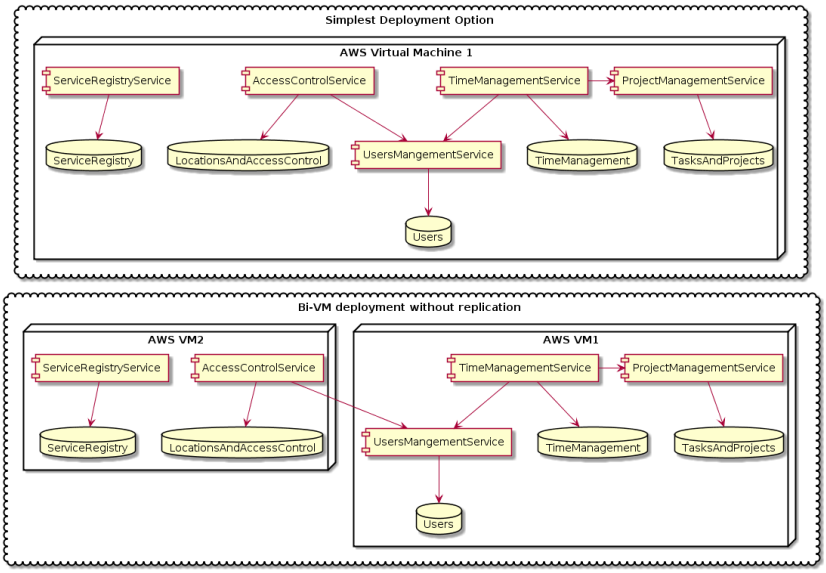

Figura 1. Dos opciones de despliegue para nuestro ejemplo

i está despegado en la MV j, para cada valor de i, al menos una variable tiene un valor de 1, es decir, todo microservicio se despliega en al menos una MV. Las variables binarias $y_{i,j}$ indican si la BD asociada al microservicio i está desplegada en la MV j, para cada i, solo una variable tiene un valor de 1, es decir, la BD de cada servicio se despliega en una única MV. Las variables $z_j \in C$ indican la configuración concreta de la MV j. GT indica el límite de utilización de las MVs a partir del cual se aplicará una regla de crecimiento de la infraestructura (se instanciará una nueva VM y se desplegarán nuevas instancias de los microservicios con más carta), y ST que indica el límite de utilización de las MVs a partir del cual se aplicará una regla de reducción de la infraestructura (se destruirá una VM). Obviamente, $0 \leq ST < GT \leq 100$, donde 100 significa una utilización máxima de la potencia de computación de las VMs instanciadas actualmente. $MaxMV$ es el límite máximo de MVs que se pueden desplegar para una configuración.

Dado un conjunto A de ANSs acordados con distintos usuarios, y una carga de trabajo esperada ω para un periodo de tiempo concreto (que especifica el volumen de peticiones que se realizarán en cada instante para cada servicio y el ANS concreto que corresponde a cada una), un conjunto interesante de objetivos a optimizar para una instancia de este problema sería:

- **Minimización del coste total de la infraestructura** $D(\omega, x, y, z)$. Este objetivo pretende minimizar el coste total del despliegue calculado como el sumatorio del coste de cada MV, que depende a su vez de la configuración de dichas VMs.

- **Minimización del número de puntos únicos de fallo** $F(x, y, z)$. Si un microservicio no está replicado, ese único despliegue supone un punto único de fallo para la lógica de la aplicación. Este objetivo pretende minimizar el número de puntos de este tipo sobre todo el conjunto de microservicios, para hacer la aplicación lo más robusta posible.
- **Minimización del tiempo de respuesta medio** $T(\omega, x, y, z)$. Dependiendo del número de despliegues, la configuración de las MVs y el perfil de carga de peticiones de cada servicio, tendremos un tiempo medio de repuesta para las peticiones. Este objetivo pretende minimizar dicho tiempo.
- **Minimización del número de violaciones de los ANS** $V(A, \omega, x, y, z)$. Dado un conjunto de ANSs con los distintos usuarios, es posible simular el comportamiento del sistema bajo un determinado perfil de carga de peticiones por usuario. Esto permitiría calcular el número de peticiones para las que se han incumplido los ANS.
- **Minimización del número de reconfiguraciones en la arquitectura** $R(A, \omega, x, y, z)$. Una reconfiguración es la instanciación o parada de una MV, con los correspondientes despliegues de servicios y BDs. Pueden existir limitaciones o costes asociado a las reconfiguraciones.

Establecer el equilibrio apropiado entre objetivos contrapuestos como el coste y el tiempo de respuesta va a depender en gran medida de las preferencias de los usuarios. Además, dado que cada MV puede tener asociado un gran número de configuraciones (más de 17.000 para Amazon [3]), el espacio de búsqueda resultante es enorme. Por ello, planteamos la necesidad de abordar este problema de optimización multi-objetivo mediante técnicas de búsqueda heurísticas. En concreto, los algoritmos evolutivos multi-objetivo (MOEAs) nos permitirán explorar de manera eficiente el espacio de configuraciones, estableciendo diferentes compromisos entre los criterios de calidad. En cuanto al trabajo futuro, pretendemos estudiar las características del problema para aplicar MOEAs, ya que puede ser necesario aplicar técnicas avanzadas dado el número de objetivos definidos. Para ello utilizaremos datos reales de Amazon, y el simulador CDOSim [1].

Referencias

1. Fittkau, F., Frey, S., Hasselbring, W.: Cdosim: Simulating cloud deployment options for software migration support. In: 6th IEEE Int. Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems. pp. 37–46 (2012)
2. Fulton, S.M.: What led amazon to its own microservices architecture. Web Page (2015), `http://thenewstack.io/led-amazon-microservices-architecture/`
3. García-Galán, J., Trinidad, P., Rana, O.F., Ruiz-Cortés, A.: Automated configuration support for infrastructure migration to the cloud. Future Generation Computer Systems 55, 200–212 (2016)
4. Mauro, T.: Adopting microservices at netflix: Lessons for architectural design. Blog entry, `https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/`
5. Shoup, R.: Service architectures at scale: Lessons from google and ebay. Talk at InfoQ.com, `http://www.infoq.com/presentations/service-arch-scale-google-ebay`