Tarea temas 4 - 6

Román Larrosa Lewandowska

ETSIIT, Universidad de Granada (UGR)
romanlarrosa@corre.ugr.es.com

Resumen. Tarea asociada a los temas 4, 5 y 6 de la asignatura Desarrollo de Sistemas Distribuidos, cursada en el segundo cuatrimestre del curso tercero del grado de Ingeniería informática en la UGR.

1 Introducción

Durante las siguientes líneas se explicará como migrar un sistema con un diseño orientado a servicios a un sistema con un diseño basado en microservicios. Se explicarán los requisitos, criterios y decisiones de diseño que habrían de aplicarse atendiendo a ciertos paradigmas del diseño de microservicios. Para ello, primero hay que definir brevemente en que se basan los diseños anteriormente mencionados y cuáles son sus diferencias.

2 Diseño orientado a servicios

También conocido como SOA (Service-Oriented Architecture). Es un estilo de diseño de software por el cual un servicio es puesto a disposición para ser consumido por otras aplicaciones de manera remota. Representa una actividad de negocio. Los principios de SOA más relevantes en nuestro contexto son:

- Contrato de estandarización: el servicio se adhiere a un estándar de comunicación, definido de manera colectiva.
- Autonomía: la relación entre servicios es minimizada. Los servicios SOA únicamente se conocen a sí mismos
- Transparencia en la localización: Los servicios deben ser accedidos desde cualquier parte de la red en la que se encuentran.
- Abstracción: Los servicios actúan como caja negra, ocultando su lógica a los consumidores

3 Diseño basado en microservicios

Los microservicios son una variante de SOA que componen una aplicación como el conjunto de varios servicios poco acoplados. En esta arquitectura, cada

microservicio tiene una tarea muy específica dentro del conjunto. Usando este paradigma de arquitectura se consiguen unos beneficios muy claros como son:

- Modularidad: los microservicios son fáciles de entender, desarrollar y testar, haciendo de esta una característica clave a la hora de optar por este paradigma en lugar de un sistema monolítico (SOA).
- Escalabilidad: como los microservicios están desplegados de manera independiente, esto es, ejecutándose en procesos diferentes, estos son fácilmente escalables.
- Desarrollo distribuido: la gestión del desarrollo de microservicios es muy sencilla puesto que una vez definidos los distintos procesos necesarios pueden dedicarse equipos de desarrollo independientes al desarrollo de cada uno de ellos, dando lugar a una paralelización del desarrollo.

Además, los microservicios aportan una gran ventaja y es que el cambio de una arquitectura monolítica a una arquitectura basada en microservicios puede hacerse de manera gradual, es decir, no necesariamente cuando un SOA monolítico quiere ser actualizado para usar microservicios debe hacerse de manera que toda su funcionalidad se despliegue en microservicios, si no que estos pueden ir desarrollándose y desplegándose en el tiempo mientras el servicio monolítico sigue en uso y a su vez haciendo uso de los nuevos microservicios.

Sin embargo, los microservicios también suponen una serie de riesgos que hay que tener en cuenta cuando nos decidimos por el uso de esta arquitectura.

- El uso de servicios independientes puede generar una barrera de información, generando un problema de comunicación
- Las llamadas entre los distintos servicios pueden generar una saturación excesiva en la red, produciendo una latencia mayor que con sistemas monolíticos.
- Aunque el testeo y la prueba de los sistemas independientes se convierte en una tarea más sencilla, el testeo del conjunto de todos los microservicios puede ser una tarea muy compleja.
- La gestión de responsabilidades entre los distintos microservicios puede tornarse muy compleja.
- El desarrollo y el soporte puede ser un problema si el equipo de desarrolladores no es fijo o se desarrollan con tecnologías diferentes.

A causa de estos problemas, antes de desarrollar un sistema basado en microservicios o plantearse migrar un sistema monolítico a una arquitectura con este paradigma es importante considerar estos problemas y plantearse los problemas derivados que pueden surgir de su uso, tales como la latencia en la red de los microservicios, el formato de los mensajes que los microservicios van a intercambiar, el balance de la carga de cada microservicio y la tolerancia a fallos de los mismos.

Además, es importante pensar si la funcionalidad que desarrolla nuestro servicio monolítico es lo suficientemente desligables, es decir, si es posible aislar las diferentes funcionalidades que desarrolla nuestro sistema monolítico sin que las dependencias

entre estas sean muy acusadas y por tanto aumenten la latencia y la carga en la red de sobremanera.

4 Problemas en la migración de monolítico a microservicios

Cuando estamos pensando en migrar nuestro sistema monolítico a un servicio basado en microservicios, los problemas que pueden surgir son numerosos pero es fácil implementar ciertos requisitos para disminuirlos drásticamente. Aunque el desarrollo de los diferentes componentes sea algo más sencillo, el despliegue de estos, la interoperabilidad y el diseño y despliegue de su infraestructura se convierte en el principal foco de problemas.

Martin Fowler [2] establece una serie de competencias básicas para disipar estos problemas las cuales tener en cuenta antes siquiera de comenzar el desarrollo de nuestro sistema de microservicios. Algunas de estas competencias son:

- Un aprovisionamiento de recursos muy ágil: al necesitar de varios sistemas para establecer nuestros microservicios, Fowler recomienda ser capaces de establecer una gran automatización con el fin de poder establecer varios servidores de manera muy rápida (en su blog menciona ser capaz de configurar un servidor en un par de horas).
- Monitoreo básico: con el uso de varios servicios acoplados colaborando se vuelve común equivocarse en las vías en las que se determinan los entornos de testeo del servicio. Esto da lugar a medidas erróneas contextualmente de la carga, el uso o la velocidad de los diferentes microservicios o redes. Con el uso del paradigma de microservicios es tan importante saber dónde se producen los errores físicos como los funcionales, y Fowler recomienda tener un sistema sencillo mediante el que ejecutar un “rollback” en caso de producirse un problema repentino que comprometa la integridad del sistema.
- Desarrollo ágil: con varios servicios que gestionar, hay que ser capaces de desplegarlos de manera rápida tanto para testarlos como para comenzar la producción. Este despliegue puede realizarse con una intervención manual en las etapas mas tempranas del desarrollo pero la máxima esperable es conseguir su automatización absoluta.

Por tanto, ¿cuándo debemos hacer la migración a microservicios de nuestro servicio monolítico? Tarde o temprano, si nuestro servicio goza de éxito, nuestro sistema monolítico llegará a su punto máximo de saturación en el que escalar el servicio se convierte en una acción demasiado costosa y el planteamiento del mismo sistema pero basado en microservicios se convierte en una alternativa mucho más escalable, modularizada y flexible.

5 Puntos clave de la migración

Para realizar la migración, normalmente se usa un modelo cíclico repetitivo, en el cual las distintas funcionalidades son desligadas del sistema monolítico en fases que se repiten hasta llegar a un resultado final en el que nuestro servicio se encuentra completamente desligado en una arquitectura basada en microservicios.

5.1 Fases de “desligado” de un servicio monolítico

El primer paso cuando queremos migrar nuestro sistema monolítico a uno basado en microservicios es encontrar qué partes del código son disgregables, y que por tanto serán las más sencillas de separar.

En este caso, las primeras funcionalidades que encontramos en la mayoría de sistemas que se convierten en micro servicios son las autenticaciones, los sistemas de notificaciones, y ciertas lógicas inherentes a cada negocio. Esto es porque son las menos acopladas al resto de funcionalidades y que por tanto es más sencillo derivarlas a otros sistemas para eliminar la carga del servicio primo monolítico.

Las siguientes funcionalidades que hay que observar son aquellas que reportan demasiada carga al sistema y que por tanto son las más propensas a individualizar para mejorar el rendimiento de nuestro servicio. Sin embargo, dichas funcionalidades muchas veces se encuentran interconectadas o muy acopladas con el resto del sistema. Por ello antes hay que realizar un acotamiento de las interdependencias del servicio monolítico.

Hay que conseguir alcanzar la máxima autonomía de cada microservicio a fin de reducir el coste de mantenimiento y la latencia ocasionada por los cambios en el modelo de datos del servicio monolítico. Para ello a menudo hay que rediseñar ciertas partes del sistema para que las funcionalidades estén muy bien encapsuladas y sean lo más independientes posibles. En caso de no conseguirlo para todas, ciertos micro servicios deberán hacer llamadas a otros, ocasionando una mayor latencia y una carga adicional sobre la red.

6 Microservicios como contenedores

Una manera sencilla y en alza de gestionar los microservicios es mediante el uso de contenedores en lugar de sistemas físicos o máquinas virtuales. Estos contenedores desempeñan únicamente la función a la que son destinados, descargándose de responsabilidades que son independientes de cada sistema y ejecutando únicamente los procesos del microservicio.

El sistema de contenedores más popular es Docker, que nos proporciona unas características muy interesantes a la hora de desplegar microservicios:

- Aislamiento de tareas: cada microservicio se ejecuta en un contenedor, que representa una tarea independiente y aislada de manera lógica del resto.

- Soporte multilenguaje: permitiendo que cada microservicio se ejecute bajo otro lenguaje de código para aprovechar mejor unas características u otras.
- Bases de datos independientes: si en la fase de desligado de funcionalidades del sistema monolítico hemos sido capaces de desligar ciertas componentes de la base de datos, éstas pueden desplegarse también en los contenedores que ejecutan cada microservicio como bases de datos separadas e independientes, mejorando los tiempos de latencia y disminuyendo la carga de peticiones comparado con un sistema con una única base de datos centralizada.
- Automatización del monitoreo: existen herramientas para monitorear de manera automática los distintos contenedores Docker, agilizando así esta tarea que ya habíamos comentado con anterioridad y que es tan esencial para el correcto despliegue y funcionamiento de una arquitectura basada en microservicios.

7 Conclusión

La implementación de una arquitectura basada en microservicios puede ser un aspecto interesante tanto si queremos crear un servicio nuevo como si queremos migrar un servicio monolítico ya existente a este paradigma. Indudablemente nos ofrece bastantes ventajas muy importantes como la modularización y el aumento de la capacidad de escalado, que facilitan en ciertos aspectos la labor de los desarrolladores a la hora de crear el sistema. Sin embargo, debemos ser muy críticos a la hora de diseñarlo y de decidir si es un paradigma apto para nuestro modelo de negocio, puesto que cometer un error a la hora de su diseño puede hacernos tener que replantearlo todo desde el principio, elevando el coste que es justo lo contrario de lo que se pretende adaptando un servicio a microservicios.

Se han dado casos de empresas que migran su modelo de negocio de un servicio monolítico a uno basado en microservicios y al tiempo han de volver a desarrollar uno monolítico puesto que no se ha llevado a cabo correctamente la migración o su lógica de negocio no era apta para este tipo de paradigma.

8 Autoevaluación

1. ¿La explicación es clara y el contenido está estructurado? Si
2. ¿Queda reflejado que se han estudiado y entendido las cuestiones de diseño cliente/servidor, modelos de replicación, metodologías de desarrollo, arquitecturas dirigidas por eventos, tecnologías middleware, etc, contenidas en la bibliografía proporcionada para las clases? Si
3. ¿Queda reflejado en el trabajo que se ha buscado y analizado suficiente bibliografía adicional y se ha comparado/contrastado la información entre las diferentes referencias encontradas (incluidas en el informe) y con la bibliografía proporcionada para las clases? Si

4. ¿La conclusión final a la que se ha llegado en el informe ha sido resultado de un estudio y análisis en profundidad y ha requerido una aportación extra por mi parte? Si
5. ¿El informe realizado me ha llevado tiempo y servido para comprender la solución al problema abordado? Si
6. ¿Se ha realizado alguna aportación adicional más allá de lo que en esencia pedía este trabajo? Si, el uso de tecnologías como Docker para el despliegue de microservicios

Considero que tras el trabajo realizado merezco una puntuación de 9 o cercana a este valor.

Referencias

1. Stenberg, J. (2014, August 11). Experiences from Failing with Microservices. Retrieved from https://www.infoq.com/news/2014/08/failing-microservices/
2. Fowler, M. (2014, August 28). bliki: MicroservicePrerequisites. Retrieved from https://martinfowler.com/bliki/MicroservicePrerequisites.html
3. D. Taibi, V. Lenarduzzi and C. Pahl, "Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation," in *IEEE Cloud Computing*, vol. 4, no. 5, pp. 22-32, September/October 2017
4. G. Mazlami, J. Cito and P. Leitner, "Extraction of Microservices from Monolithic Software Architectures," 2017 IEEE International Conference on Web Services (ICWS), Honolulu, HI, 2017
5. Docker INC. Docker Docs, from https://docs.docker.com/, última visita: 30 mayo 2020.
6. Noonan, A. (2020, May 26). To Microservices and Back Again. Retrieved from https://www.infoq.com/presentations/microservices-monolith-antipatterns/