Sistemas Distribuidos

Tarea 2: Arquitectura distribuida utilizando microservicios

Ignacio Cisternas Núñez y Pablo Aguirre Morreau

January 3, 2020

Abstract

La tarea 2 de SD (INF-343) tiene como objetivo introducir el uso de una arquitectura distribuida en programación utilizando un diseño de microservicios. Se solicitan dos tipos de tecnologías diferentes: Remote procedure y Asynchronous Messaging. De acuerdo a lo realizado e investigado se presentan la comparacion entre ambas.

1 Remote Procedures

Como lo indica su nombre, llamada de procedimientos remotos (RPC). En pocas palabras se refiere al uso de mensajes para que un proceso le solicite a otro que ejecute una tarea o subrutina especificada. Es esencial que tanto el cliente que solicita la tarea como el servidor que la ejecuta compartan un protocolo de mensajes para que puedan entenderse entre sí, para esto son fundamentales los stub, cuya funcion es empaquetar los parámetros implicados en la solicitud a un mensaje comprensible por el servidor y lo mismo con la respuesta, empaquetando lo retornado por la llamada de la funcion.

Esta tegnología, entre todas sus cualidades, presenta beneficios como: ofrecer un entorno lo más similar posible a un ambiente local, lo que simplifica la comprension de codigo; evitar el trabajo directo con sockets y conexiones entre procesos; ocultar detalles de implementacion directa utilizando en cambio un protocolo de peticion / respuesta; Alta escalabilidad, concurrencia y tolerancia a fallos al distribuir la carga. En resumen, apunta a ofrecer un middleware lo mas simple posible para desarrollo distribuido.

1.1 gRPC

En esta tarea puntual, el servicio de procedimientos remotos utilizados fue gRPC. El cual es un framework de código abierto (open source) inicialmente desarrollado por Google que utiliza HTTP/2 como protocolo de transporte y Protocol Buffers como lenguaje de interfaz entre los procesos que desean comunicarse. Una de las ventajas de gRPC es que soporta la interacción entre códigos de distintos lenguajes como c++, Java, Python, Go, Ruby, entre varios otros.

2 Asynchronous Messaging

La mensajería asíncrona (Asynchronous Messaging) plantea como idea principal el envío y recepcion de mensajes sin la necesidad de una presencia de ambas partes para que estos sean efectivos. Esto aporta muchos beneficios sobre todo en rendimiento, pues al no requerirse una respuesta a cada mensaje enviado, es posible continuar con otras tareas inmediatamente despues del envío. Lo principal es contar con un broker de mensajería que funcione como un buzon de mensajes, es decir, almacene los mensajes correspondientes a cada usuario en el tiempo en el que estos no se encuentren dosponible, desplegandolos una vez que estos vuelvan a conectarse.

2.1 RabbitMQ

RabbitMQ es un software de middleware desarrollado por Rabbit Technologies Ltd. Permite protocolos como HTTP, XMPP, entre otros y Mozilla Public License son los dueños actuales de su licencia. El objetivo de este software es ser un intermediario entre clientes y servidor mediante message brokers lo que implica que estos no se conozcan entre sí. Simplemente se comuniquen con sus respectivos exchanges que se encargan de encolar adecuadamente cada mensaje.

3 Conclusión

Luego de lo explicado y lo experimentado en esta tarea, hay varios puntos a destacar sobre el uso de ambas tegnologías.

El objetivo principal de la aplicacion o servicio a desarrollar es lo más importante al momento de escoger cual opción utilizar. Si lo que buscamos es principalmente un sistema de chat o mensajería (como el caso de esta tarea). Asynchronous Messaging es una excelente opción por la optimización de tiempo y recursos de cada máquina. RabbitMQ no requiere tanta complejidad ni estructura en el paso de mensajes comparado con gRPC, el cual exige definir la composición exacta de los servicios y mensajes involucrados en la comunicación. Además, el sistema de colas directamente ofrece almacenamiento de datos para cuando los usuarios no se encuentran activos. Esto es trabajo del desarrollador para el caso de los procedimientos remotos

RPC es conveniente de utilizar para aquellas aplicaciones que requieren de procesos más complejos. Tanto en su procesamiento como en un mayor número de variables involucradas, pues se dispone de un paso de parámetros y retorno más sofisticado que ofrecen un mayor y más ordenado traspaso de información entre cada transacciíon.

Es completamente posible utilizar RPC para cualquier servicio que ya se haya implementado en RabbitMQ, sin embargo, la mayor simplicidad de comprensión y el menor trabajo para el desarrollador involucradas en esta última tecnología nos hacen concluir que de ser posible, es preferente utilizar Asynchronous Messaging. Especialmente si se trata de applicaciones de mensajería o chat.