Informe Comparativo
Tarea 2
Sistemas Distribuidos

Leonardo Astudillo - Rol: 201573598-0
Claudio Valenzuela - Rol: 201573530-1

Parte 1:
gRPC es un framework del protocolo Remote Procedure Call (RPC), de código abierto, creado por Google, para conectar microservicios. La arquitectura RPC se basa en que el cliente envía una solicitud para que el servidor la procesa y de esa manera enviar la respuesta de vuelta al cliente. Esta arquitectura no tiene un foco en las comunicaciones entre ambas máquinas.

La principal ventaja que tiene esta herramienta es primero la manera sencilla de establecer el formato de la data y los rpc asociados al servicio. Dando la flexibilidad de crear clientes y servidores con lenguajes de programación distinto

Otra ventaja es el hecho de que gRPC está focalizado en el rendimiento del servicio, gracias a la estructura del formato protocol buffer, que es la manera de serializar data estructurada

Además tiene la posibilidad de establecer un protocolo bidireccional, cliente-servidor. De tal manera que tanto el cliente como el servidor permite enviar streaming de datos de manera que permite tanto sincronicidad como asincronicidad en los servicios.

Para el caso de la tarea, se observó una desventaja importante, ya que a pesar de haber asincronicidad en los mensajes, el servidor no puede establecer conexión con el cliente para enviar el mensaje de otro cliente, lo que genera un conjunto de problemas, uno de ellos era establecer alguna ID única con los clientes. Debido a que la manera con que se obtiene la ID, que es a partir del IP y puerto del cliente. Lo que podría generar colisiones si es que alguien tuviera la misma IP y puerto. El otro problema observado es el hecho de que todos los clientes deben establecer un servicio continuo, debido a la arquitectura de los RPC, solicitud - respuesta. Que no termina hasta que el cliente se desconecta. Lo que podría generar un malgasto de recursos del cliente y para el caso del servidor una sobrecarga del servicio de streaming, que se utiliza para la transmisión de mensajes. Finalmente está el caso, de que el cliente tiene que estar activo para recibir mensajes, si el cliente no está activo, no va a recibir el mensaje

Parte 2: RabbitMQ es un software de negociación de mensajes que entra dentro de la categoría de middleware. Este implementa el protocolo de estándar abierto AMQP(Advanced Message Queuing Protocol). RabbitMQ corresponde a un broker de mensajería lo que significa que es un middleware orientado a mensajes, es decir, es un agente que se encarga de la transferencia de mensajes, esto lo hace mediante el uso de

colas donde las aplicaciones pueden conectarse a estas y leer o transferir mensajes en estas.

Un broker de mensajería proporciona asincronicidad al sistema, ya que una vez que el cliente envía un mensaje este puede realizar otro trabajo o enviar mas mensajes, ya que el broker es el encargado de proporcionar el mensaje a un cliente receptor. El broker además proporciona almacenamiento persistente para hacer una copia de seguridad de la cola de mensajes, esto significa que el emisor y el receptor de los mensajes no necesitan estar conectados en la red al mismo tiempo, resolviendo así problemas como conectividad intermitente, en efecto, si el receptor llegase a fallar los mensajes se acomulan en la cola para su posterior prosesamiento.

Los broker de mensajería ofrecen ventajas como la escalabilidad y el rendimiento, ya que las colas pueden balancear la respectiva carga, flexibilidad a la arquitectura, persistencia en la entrega de mensajería y garantizan la entrega y el orden. Por otra parte las desventajas que tiene este sistema es la agregación de este componente adicional que supone un costo mayor en el mantenimiento y complejiza el sistema. También puede ser más complicado implementar un sistema síncrono con este método.

Para el caso de la tarea, el uso del software RabbitMQ funcionó de buena manera puesto que lo que se solicitaba era un traspaso de mensajería entre aplicaciones con un servidor de intermediario, tarea en la que se especializa un middleware orientado a mensajes(MOM), por lo que no existen mayores problemas en el funcionamiento del sistema, es más, se podría haber implementado de manera más simple que lo estipulado en la tarea, sin la existencia de un servidor como intermediario, creando una conexión directa entre clientes con las colas, lo que haría la función del programa aún más eficiente con una arquitectura mucho más simple.

Conclusión:

Ambas opciones tienen el mismo rendimiento para el caso en donde solo hay 2 clientes. Pero para el caso de n clientes, es mejor utilizar mensajes asíncronos junto con el broker de mensajera RabbitMQ, debido a que los mensajes tienen persistencia cuando el cliente o servidor está inactivo. Dado que el protocolo RPC es esencialmente síncrono mientras que el protocolo MOM es esencialmente asíncrono (aunque ambos protocolos pueden implementarse de ambas maneras), es mejor recomendar RabbitMQ para este tipo de tareas como se estipulo anteriormente, puesto que ya tiene de forma base lo que requiere un menor esfuerzo en la implementación del traspaso de mensajes estipulado en la tarea.