Informe comparativo gRPC vs RabbitMq

Pedro Chacon, rol: 201473561-8
Daniel Pacheco, rol: 201404570-0

Resumen:

En el presente informe se realiza una comparación de dos frameworks, gRPC y RabbitMQ, con el objeto de discernir cuál de ellos resulta más conveniente para implementar un servicio de chat.

Comparación de gRPC vs RabbitMQ:

RabbitMQ:

RabbitMQ es un broker de mensajería de código abierto, distribuido y escalable, que sirve como intermediario para la comunicación eficiente entre productores y consumidores.
En RabbitMQ se definen colas que van a almacenar los mensajes que envían los productores hasta que las aplicaciones consumidoras obtienen el mensaje y lo procesan. Con lo experimentado a lo largo del desarrollo de esta tarea se pudo observar una serie de pros y contras al momento de usar RabbitMQ.

Antes de empezar a usar RabbitMQ es necesario la instalación de un par de softwares extra que son necesarios para el funcionamiento de este broker. Se necesita instalar el servidor de RabbitMQ, el que se encargará de recibir nuestros mensajes y distribuirlos de la manera que nosotros determinemos, este servidor está hecho sobre el lenguaje Erlang, por lo que es necesario instalar este lenguaje para el correcto funcionamiento del servidor. Cabe destacar que para el caso de esta tarea se utilizó el lenguaje de programación Python, el cual trabaja con la librería Pika para interactuar con RabbitMQ. De esta manera antes de empezar a generar nuestra arquitectura de micro servicios con Rabbit se requiere descargar estos tres elementos.
El servidor que provee RabbitMQ es robusto, maneja bien la tolerancia a fallos y la fiabilidad, además de entregar una serie de diferentes tipos de envio de mensaje, con esta variedad permite que sea muy flexible y pueda adaptarse a distintos problemas. Esto nos permitió que la comunicación entre clientes y servidor sea muy estable, al punto que mientras el nombre de las colas sean las correspondientes se pueda mandar mensajes a clientes que no se encuentren conectados. Como se mencionó previamente, se utilizó la librería Pika para la interacción entre Python y RabbitMQ, con esta se manejó la conexión al servidor, el envío y recepción de mensajes, Pika hace un buen trabajo manteniendo simple el uso de estas funcionalidades.
Dentro de los problemas que encontramos con RabbitMQ (o como Pika implementa esta interacción) es el recibo de mensajes, básicamente cada vez que se recibe un mensaje este mensaje es procesado por una función que lo procesa en base a eso se puede generar una ilusión de una arquitectura de micro servicios, pero en realidad no lo es, esto debido a que solo una misma función se encarga de todos las solicitudes a los servicios en vez de tener cada uno por separado.

gRPC:

gRPC es un framework que permite el intercambio de mensajes entre dos aplicaciones, esto lo hace estableciendo una conexión entre ambas, permitiendo que la aplicación que actúa a forma de cliente pueda realizar la invocación de procedimientos en la aplicación que se porta como servidor. El proceso de la invocación de un procedimiento se realiza enviando un mensaje y recibiendo otro a cambio.

Debido a la implementación del framework, los desarrolladores solo deben preocuparse de implementar los procedimientos tanto en el cliente como en el servidor, sin necesidad de tener en cuenta el cómo es que los mensajes llegan al destinatario.

Uno de los puntos importantes a destacar de gRPC es que si bien no requiere la instalación de complejos softwares adicionales, siendo tan solo necesario la instalación de una librería para el lenguaje en el cual se usará el framework (python en este caso), gRPC sí requiere del uso de un lenguaje adicional, llamado protobuf, aunque cabe destacar que no es de gran complejidad y se puede aprender fácilmente.

Protobuf es el lenguaje mediante el cual se definen los tipos y estructuras de los datos que serán enviados en el intercambio de mensajes, esto puede incurrir en ventajas o desventajas dependiendo de la situación, ya que con una estructura definida de los datos, se permite la detección tanto al envío como en la recepción de que los mensajes están en si su correcto formato, lo que ayuda en la detección de errores. Lo planteado se puede volver una desventaja si se considera las aplicaciones pueden requerir ser reformuladas en un futuro, lo que implica que se deben modificar las definiciones realizadas en el lenguaje protobuf, por tanto se debe mantener una correcta documentación de los parámetros para las estructuras de datos definidas.

Conclusión:

Luego de exponer las ventajas y desventajas de cada uno de los frameworks utilizados, en primera instancia se concluye que RabbitMQ sería la opción ideal para implementar un chat, debido a lo robusto que es este broker de mensajería, además que Pika entrega una manera muy simple de utilizar las funcionalidades que provee el servidor de RabbitMQ, entregando gran libertad al momento de desarrollar lo que sea necesario para solucionar un problema.

Pero si además de solo considerar el hecho de construir un chat, se toman otros factores como los alcances de la tarea y el enfoque a micro servicios que se le da, es que gRPC se vuelve una alternativa que permite construir una herramienta de chat en pocos pasos, con un lenguaje de interfaz que no llega a ser complejo y sin necesidades de instalar gran cantidad de software extra.

Por lo tanto se concluye que para una implementación como la que se pide en la tarea ambos frameworks se vuelven una alternativa viable, con sus pros y contras. En cambio si el objetivo es crear un chat a gran escala, definimos a RabbitMQ como mejor framework para este propósito.