Informe Comparativo entre gRCP y RabbitMQ

Fabian Riquelme Macías
Sebastian Ávalos

Contexto

El problema consistía en lograr levantar un servicio de mensajería tipo "chat" (no-live), con un único servidor que lograse dar soporte a **N** clientes. El servicio debía ser desarrollado en lenguaje Python tanto con el framework gRCP (Remote Procedure Call) como con RabbitMQ (Broker).

Donde las funcionalidades principales que debía ofrecer el programa consistían en:

- Tener un servicio que permitiese enviar mensajes
- Tener un servicio que permitiese ver los mensajes enviados
- Tener un servicio que permitiese ver los mensajes recibidos por otros usuarios
- Tener un servicio que permitiese al usuario ver a los otros usuarios bajo el nombre por el cual se les puede enviar correos
- Tener un servicio que proporcionase identificación, y que el cliente fuese único
- Mantener un log sobre los mensajes enviados/recibidos + TimeStamp

1 Implementación mediante gRCP(1)

gRpc es un framework que ofrece una implementación del protocolo RCP (Remote Procedure Call), el cual como muchos otros sistemas que implementan RCP, está basado en la idea de definir un servicio, especificar los métodos que pueden ser llamados remotamente con sus parámetros y retornos; para que de esa forma tales métodos que normalmente están alojados en el servidor puedan ser accedidos de manera local para el cliente. gRCP usa Protocol Buffers como Lenguaje de Definición de Interfaces (IDL) para describir tanto la estructura como la carga del mensaje.

En palabras simples gRCP permite mediante un archivo '.proto' la definición de objetos y métodos a utilizar independiente de su implementación. Por lo que finalmente entidades puede utilizar métodos no disponibles localmente como si lo estuviesen.

2 Implementación mediante RabbitMQ(2)

RabbitMQ es un Broker de mensajería que implementa un sistema de servicio asíncrono de mensajes mediante el uso de colas, donde normalmente un agente actúa como productor y otro como consumidor. La mejor analogía es que funciona como una oficina de correos, donde uno deja correo en una caja de correspondencia y podemos asegurar que llegará a su destino.

Una cola representa nada mas que un gran buffer de mensajes, un productor alguien que envía mensajes, y un consumidor alguien que en su mayor parte se dedica a recibir mensajes. Pero a diferencia de

la arquitectura clásica de cliente-servidor los mensajes pasan siempre a través de la cola de mensajes ofrecida por el Broker.

La implementación se realizó mediante la librería "Pika" y el uso de colas "Request" y "Reply" que representan respectivamente los envíos y respuestas. La librería Pika ofrece muchas opciones sobre configuración del Broker, tipos de mensajes e incluso la posibilidad de utilizar Acknowledgement (ACK), por lo que ofrece características de bajo nivel.

3 Contraste entre gRCP y RabbitMQ

- En términos de simplicidad de implementación, ambos son muy sencillos de llevar a cabo, donde en general el nivel de configuración previa es en mismas cantidades, a pesar de que gRCP necesite la configuración de la interfaz en IDL
- En términos de transparencia, gRCP vence RabbitMQ, debido a que necesita la definición de una única interfaz ".proto" que puede ser compilada y utilizada por cualquier lenguaje de programación.
- En términos del desarrollo, gRCP puede ser utilizado para generar "Stub", interfaces con métodos sin desarrollar o parcialmente desarrollado. Y por ende cuando los métodos sean desarrollados el cliente no deberá cambiar nada.
- En términos del contexto de la tarea, gRCP consiste en llamadas remotas, por ende es necesario aplicar una función para obtener los mensajes, mientras que RabbitMQ administra la cola por si mismo
- En términos de configuración, RabbitMQ ofrece una mayor libertad de configuración en sus campos (como antes se vió, uno de ellos es el ACK), por ende permite el desarrollo de servicios mas especializados.

4 Conclusión

La implementación mas prometedora parece ser gRCP, ya que como se está implementando un servicio no-live no es necesario una actualización continua de mensajes. Por ende una actualización discreta y desencadenada por una función es ideal. Mientras que si el trabajo a realizar hubiese sido un chat live, una cola siempre en escucha hubiera sido mucho mas eficiente para manejar las colas de mensajes entre múltiples usuarios.

En términos de escalabilidad ninguno sobresale del otro, gRCP está limitado por la capacidad de computo del servidor(El cual puede verse solucionado por el uso de clusters), mientras que RabbitMQ está delimitado por el tamaño del disco y el buffer de la cola de mensajes, sin mencionar la sobre carga de trabajo debido al orquestar mensajes. Finalmente es curioso mencionar que mediante RabbitMQ es posible implementar RCP orientado a mensajes

References

[1] https://grpc.io/docs/guides/concepts/1905.

[2] https://www.rabbitmq.com/tutorials/tutorial-one-python.html