**¿Por qué gRPC?**

gRPC es un marco moderno de llamada a procedimiento remoto (RPC) de código abierto y alto rendimiento que puede ejecutarse en cualquier entorno. Puede conectar de manera eficiente servicios dentro y entre centros de datos con soporte conectable para equilibrio de carga, rastreo, verificación de estado y autenticación. También es aplicable en la última milla de la informática distribuida para conectar dispositivos, aplicaciones móviles y navegadores a servicios backend.

**gRPC le permite definir cuatro tipos de métodos de servicio**:

* RPC unarios donde el cliente envía una única solicitud al servidor y obtiene una única respuesta, como una llamada de función normal.

**rpc** SayHello(HelloRequest) **returns** (HelloResponse);

* RPC de transmisión del servidor donde el cliente envía una solicitud al servidor y obtiene una secuencia para leer una secuencia de mensajes. El cliente lee la secuencia devuelta hasta que no hay más mensajes. gRPC garantiza el orden de los mensajes dentro de una llamada RPC individual.

**rpc** LotsOfReplies(HelloRequest) **returns** (stream HelloResponse);

* RPC de transmisión del cliente donde el cliente escribe una secuencia de mensajes y los envía al servidor, nuevamente utilizando una secuencia proporcionada. Una vez que el cliente ha terminado de escribir los mensajes, espera a que el servidor los lea y devuelva su respuesta. Nuevamente, gRPC garantiza el orden de los mensajes dentro de una llamada RPC individual.

**rpc** LotsOfGreetings(stream HelloRequest) **returns** (HelloResponse);

* RPC de transmisión bidireccional donde ambas partes envían una secuencia de mensajes utilizando una secuencia de lectura y escritura. Los dos flujos operan de forma independiente, por lo que los clientes y servidores pueden leer y escribir en el orden que quieran: por ejemplo, el servidor podría esperar a recibir todos los mensajes del cliente antes de escribir sus respuestas, o alternativamente podría leer un mensaje y luego escribir un mensaje. o alguna otra combinación de lecturas y escrituras. Se conserva el orden de los mensajes en cada secuencia.

**rpc** BidiHello(stream HelloRequest) **returns** (stream HelloResponse);

**¿Cuál es la diferencia entre gRPC y REST?**

gRPC y REST son dos formas de diseñar una API. Una API es un mecanismo que permite a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos. En gRPC, un componente (el cliente) llama o invoca funciones específicas en otro componente de software (el servidor). En REST, en lugar de llamar a funciones, el cliente solicita o actualiza datos en el servidor.

**¿Qué es gRPC?**

gRPC significa Google Remote Procedure Call. gRPC es un marco RPC de código abierto que se utiliza para crear APIs escalables y rápidas. Permite el desarrollo de sistemas en red y la comunicación abierta entre las aplicaciones cliente y servidor de gRPC. gRPC ha sido adoptado por varias empresas tecnológicas de primer nivel, como Google, IBM y Netflix, entre otras. El marco de trabajo de gRPC depende de pilas de tecnología de vanguardia como HTTP/2, búferes de protocolo y más para una óptima protección de la API, llamadas a procedimientos remotos de alto rendimiento y escalabilidad.

Diagrama

Descripción generada automáticamente

**¿Qué es RPC?**

RPC y REST - Representational State Transfer- han sido históricamente dos enfoques distintos para construir APIs. Además, protocolos como SOAP y GraphQL también se utilizan para este fin. Las llamadas a procedimientos remotos permiten escribir software como si se fuera a ejecutar localmente, aunque se ejecute en un dispositivo diferente.

Son el marco más utilizado para diseñar APIs. A diferencia de la típica llamada al protocolo HTTP, una RPC emplea una llamada a una función como método principal de interacción entre el cliente y el servidor. RPC es una técnica productiva para crear APIs, ya que los intercambios son sencillos y los contenidos son ligeros. Los servicios gRPC también imitan esta arquitectura de comunicación. RPC emplea IDL - Interface Definition Language para contratar el tipo de datos y los métodos que serán invocados. Los servicios gRPC adoptados a partir de los RPC se han hecho muy populares en los últimos años.

**Búferes de protocolo**

Los búferes de protocolo, también conocidos como Protobuf, son estándares de serialización o deserialización que simplifican la definición de aplicaciones y realizan automáticamente la generación de código de las bibliotecas de clientes. La versión más reciente, proto3, es más sencilla de utilizar y ofrece las capacidades más novedosas para gRPC..Proto Los archivos habilitan los servicios gRPC y las comunicaciones entre los clientes gRPC y los mensajes del servidor. El archivo .proto es cargado en memoria en la ejecución por el compilador Protobuf - protoc. Este compilador construye aplicaciones de cliente gRPC y de servidor gRPC que emplean la estructura en memoria para serializar y deserializar los datos binarios. Cada comunicación se envía y recibe entre el usuario y el servicio remoto siguiendo la generación de código en gRPC.Como los datos se traducen a una forma binaria y las señales codificadas son más pequeñas, el análisis sintáctico con Protobuf utiliza menos potencia de CPU para gRPC. Por lo tanto, incluso en ordenadores con un CPU más débil, como los teléfonos móviles, los mensajes se envían más rápidamente con gRPC.

**HTTP/2**

El servicio gRPC está construido sobre HTTP/2, la versión de HTTP/1.1 que tiene menos limitaciones. Aunque funciona con el protocolo HTTP más antiguo, HTTP/2 tiene varias características sofisticadas para gRPC. Esto incluye una capa de encuadre binario, que divide cada consulta y respuesta de HTTP/2 en mensajes más pequeños y los encuadra en formato binario para mejorar la entrega del mensaje. Además, gRPC soporta múltiples peticiones y respuestas del cliente y del servidor gRPC en flujo bidireccional full-duplex.

HTTP/2 cuenta con un método de control de flujo que permite un control preciso de la RAM necesaria para almacenar paquetes en vuelo. También ofrece compresión de cabeceras para los servicios gRPC. Todo en HTTP/2 se encripta antes de la transmisión, incluso las cabeceras, lo que proporciona un alto rendimiento en las llamadas a procedimientos remotos. gRPC proporciona un procesamiento tanto asíncrono como síncrono con HTTP/2, permitiendo la ejecución de varios tipos de RPC interactivos y en streaming.

Con la ayuda de todas estas características de HTTP/2, los servicios gRPC pueden utilizar menos recursos, lo que conduce a tiempos de respuesta más rápidos entre las aplicaciones basadas en la nube y los servicios gRPC y una mayor duración de la batería para los clientes gRPC que operan en dispositivos móviles.

**Streaming**

Una idea crucial que soporta gRPC es el streaming, que permite la ejecución de varios procesos dentro de una única solicitud. gRPC hace esto posible a través de la función de multiplexación de HTTP/2, que permite enviar o recibir varias respuestas o solicitudes simultáneamente a través de una conexión TCP - Transmission Control Protocol -. Los principales formatos de streaming son los RPCs server-streaming, client-streaming RPCs, y bidirectional-streaming RPCs.

**Canales**

A diferencia de los streams de HTTP/2, que permiten numerosos streams simultáneos en una sola conexión de solicitud, un canal con gRPC soporta múltiples streams continuos a través de múltiples solicitudes. Se emplean para construir un stub de cliente y dan un mecanismo para enlazar con un servidor gRPC en una IP y un puerto específicos.

**Arquitectura gRPC**

La arquitectura gRPC consiste en un cliente gRPC y un servidor gRPC. Cada servicio cliente gRPC contiene un stub o un archivo autogenerado, que es similar a una interfaz que contiene los procesos remotos activos. El cliente gRPC inicia una llamada de procedimiento local a un stub que contiene los argumentos que deben ser reenviados a los mensajes del servidor gRPC. A continuación, el stub del cliente gRPC envía la consulta a la unidad local del cliente en el ordenador local después de serializar los argumentos mediante el procedimiento de marshaling Protobuf.

El sistema operativo utiliza el protocolo HTTP/2 para comunicarse con el ordenador servidor distante. El sistema operativo del servidor acepta los mensajes e invoca el proceso stub del servidor, que utiliza Protobuf para invocar la operación apropiada después de decodificar los parámetros entrantes. La capa de transporte del cliente recibe entonces la respuesta cifrada del stub del servidor. La ejecución pasa de nuevo a la persona que llama después de que el stub del cliente gRPC reciba los mensajes de respuesta y desenvuelva los argumentos proporcionados.