gRPC

Huh?

- gRPC adalah *framework* yang digunakan untuk membangun *Remote Procedure Calls (RPC)* berkinerja tinggi.
- Artinya, gRPC memungkinkan aplikasi Anda (client) untuk memanggil fungsi pada program lain (server) yang berlokasi di komputer yang berbeda, seolah-olah fungsi tersebut berjalan secara lokal.

Keuntungan Menggunakan gRPC

- **Kinerja tinggi:** menggunakan protokol HTTP/2 untuk mencapai kinerja yang sangat cepat dan efisien.
- **Efisiensi bandwidth:** menggunakan *protocol buffers* untuk encoding dan decoding data, yang menghasilkan transfer data yang lebih kecil dan lebih efisien.
- **Skalabilitas:** dirancang untuk di-scale untuk melayani banyak client secara serempak.

Keuntungan Menggunakan gRPC

- **Dukungan multi-bahasa:** tersedia dalam berbagai bahasa pemrograman, sehingga Anda dapat membangun client dan server dalam bahasa yang Anda sukai.
- **Dukungan streaming:** mendukung streaming data, yang berguna untuk kasus-kasus seperti transfer file besar atau komunikasi real-time.

Prinsip Kerja gRPC

- **Definisi service:** Anda mendefinisikan service yang berisi fungsi-fungsi yang dapat dipanggil dari jarak jauh. Service ini didefinisikan menggunakan *protocol buffers*, yang merupakan format bahasa-agnostik untuk menggambarkan struktur data.
- **Implementasi server:** Anda mengimplementasikan fungsi-fungsi service tersebut pada server.
- **Generasi client stub:** gRPC menyediakan tools untuk secara otomatis generate kode client stub untuk bahasa pemrograman yang Anda gunakan. Client stub berisi pembungkus untuk fungsi-fungsi service tersebut, sehingga Anda dapat memanggil fungsi-fungsi tersebut dari kode Anda.

Prinsip Kerja gRPC

- **Client calls server:** Client Anda menggunakan client stub untuk memanggil fungsi pada server.
- **Server processes request:** Server memproses request dari client dan mengirimkan response kembali.

Contoh Penggunaan gRPC

- Microservices: digunakan secara luas untuk membangun microservices, di mana aplikasi dibagi menjadi layanan kecil yang berkomunikasi satu sama lain melalui gRPC.
- API: digunakan untuk membangun API berkinerja tinggi dan efisien.
- Mobile applications: dapat digunakan untuk membangun aplikasi mobile yang membutuhkan komunikasi real-time dengan server.

Contoh Pendefinisian Service

```
syntax = "proto3";

package helloworld;

service Greeter {
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}

message HelloRequest {
  string name = 1;
}

message HelloReply {
  string message = 1;
}
```

Contoh Kompilasi Service

```
# Install the protocol buffer compiler (protoc)
# https://grpc.io/docs/protoc-installation/

protoc
  --js_out=import_style=commonjs,binary:.
  --grpc_out=.
  --plugin=protoc-gen-grpc=`which grpc_tools_node_protoc_plugin`
  helloworld.proto
```

Contoh Implementasi Server

```
const { Server } = require('grpc')
const { GreeterService } = require('./helloworld_grpc_pb')
const { GreeterServiceImpl } = require('./greeter_impl')

const server = new Server()

server.addService(GreeterService, new GreeterServiceImpl())

server.bindAsync('0.0.0.0:50051', (err, port) => {
  if (err) {
    console.error(err)
    return
  }
  console.log(`Server listening on ${port}`)
  server.start()
})
```

Contoh Implementasi Client

```
const { Client } = require('grpc')
const { GreeterClient } = require('./helloworld_grpc_pb')

const client = new Client('localhost:50051', GreeterClient)

const request = { name: 'John Doe' }

client.sayHello(request, (err, response) => {
  if (err) {
    console.error(err)
  } else {
    console.log('Greeter client received:', response.message)
  }
})
```