grpc

> author: Zilin Xu
> date: 2024.6.12
> resource: grpc官方文档中文版, b站狂神说， Youtube

gRPC relies on HTTP/2 and protocol buffer

HTTP/2 是超文本传输协议 (HTTP) 的第二个主要版本，是对 HTTP/1.1 的改进。HTTP/2 的设计目标是提高性能，减少延迟，使得网络应用程序加载更快。以下是 HTTP/2 的一些关键特性：

1. **二进制分帧层**：HTTP/2 将数据传输转换为二进制格式，而不是 HTTP/1.1 的文本格式。二进制协议更加紧凑和高效，减少了数据传输的开销。
2. **多路复用**：HTTP/2 允许多个请求和响应在一个连接上同时进行，而无需等待先前的请求完成。这解决了 HTTP/1.1 中的“队头阻塞”问题。
3. **头部压缩**：HTTP/2 使用 HPACK 算法对请求和响应头部进行压缩，减少了传输数据的大小，提高了传输效率。
4. **服务器推送**：HTTP/2 允许服务器主动向客户端推送资源，而不是等待客户端请求。这可以减少延迟，加快页面加载速度。
5. **优先级和流控制**：HTTP/2 引入了对请求和响应流的优先级控制和流量控制，确保重要的资源优先传输，提高用户体验。

HTTP/2 兼容 HTTP/1.1 的所有功能，同时通过引入这些改进，显著提高了性能和效率。这使得 HTTP/2 成为现代网络应用的首选传输协议之一。

start our project: a simple chat service

create directory

```
mkdir grpc
go mod init grpc
touch server.go
```

edit server.go

```
package main

import (
    "log"
    "net"
```

```
    "google.golang.org/grpc"
)

func main() {
    //set up connection and error handling
    lis, err := net.Listen("tcp", ":9000")
    if err != nil {
        log.Fatalf("error with listening on port 9000: %v", err)
    }

    //import grpc server
    grpcServer := grpc.NewServer()

    err = grpcServer.Serve(lis)
    if err != nil {
        log.Fatalf("fail to serve gRPC server over port 9000 %v", err)
    }
}
```

make a new directory `/proto` and edit `chat.proto`

```
syntax = "proto3";

option go_package = "/grpc/proto";

message Message {
    string body = 1;
}

service ChatService {
    rpc SayHello(Message) returns (Message) {}
}
```

run following command to generate `.pb.go` and update `go.mod`

```
➜  grpc git:(main) ✗ protoc --go_out=. --go_opt=paths=source_relative --
go-grpc_out=. --go-grpc_opt=paths=source_relative proto/chat.proto


➜  grpc git:(main) ✗ go mod tidy
```