文档

源码：https://github.com/grpc/grpc-go
官方文档：https://www.grpc.io/docs/what-is-grpc/introduction/
protobuf编译器源码：https://github.com/protocolbuffers/protobuf
proto3文档：https://protobuf.dev/programming-guides/proto3/

gRPC介绍

```
gRPC 是一个高性能、开源和通用的 RPC 框架，面向移动和 HTTP/2 设计。目前提供 C、Java 和 Go 语言版本，分别是：grpc, grpc-java, grpc-go. 其中 C 版本支持 C, C++, Node.js, Python, Ruby, Objective-C, PHP 和 C# 支持.
```

gRPC 基于 HTTP/2 标准设计，带来诸如双向流、流控、头部压缩、单 TCP 连接上的多复用请求等 特性。这些特性使得其在移动设备上表现更好，更省电和节省空间占用。
在 gRPC 里客户端应用可以像调用本地对象一样直接调用另一台不同的机器上服务端应用的方法， 使得您能够更容易地创建分布式应用和服务。与许多 RPC 系统类似，gRPC 也是基于以下理念： 定义一个服务，指定其能够被远程调用的方法（包含参数和返回类型）。在服务端实现这个接口， 并运行一个 gRPC 服务器来处理客户端调用。在客户端拥有一个存根能够像服务端一样的方法。

gRPC特点

1. 语言中立，支持多种语言
2. 基于IDL（接口定义语言）定义服务U，通过proto3工具生成指定语言的数据结构、服务端以及客户端stub
3. 通信协议基于标准的 http/2 设计，支持双向流、消息头压缩、单 tcp 的多路复用、服务端推送等特性，这些特性使得 gRPC 在移动端设备上更加省电和节省网络流量；
4. 序列化支持 protobuffer 和 json，protobuffer 是一种语言无关的高性能序列化框架，基于 http/2 + protobuffer，保证了 RPC 调用的高性能；

gRPC使用场景

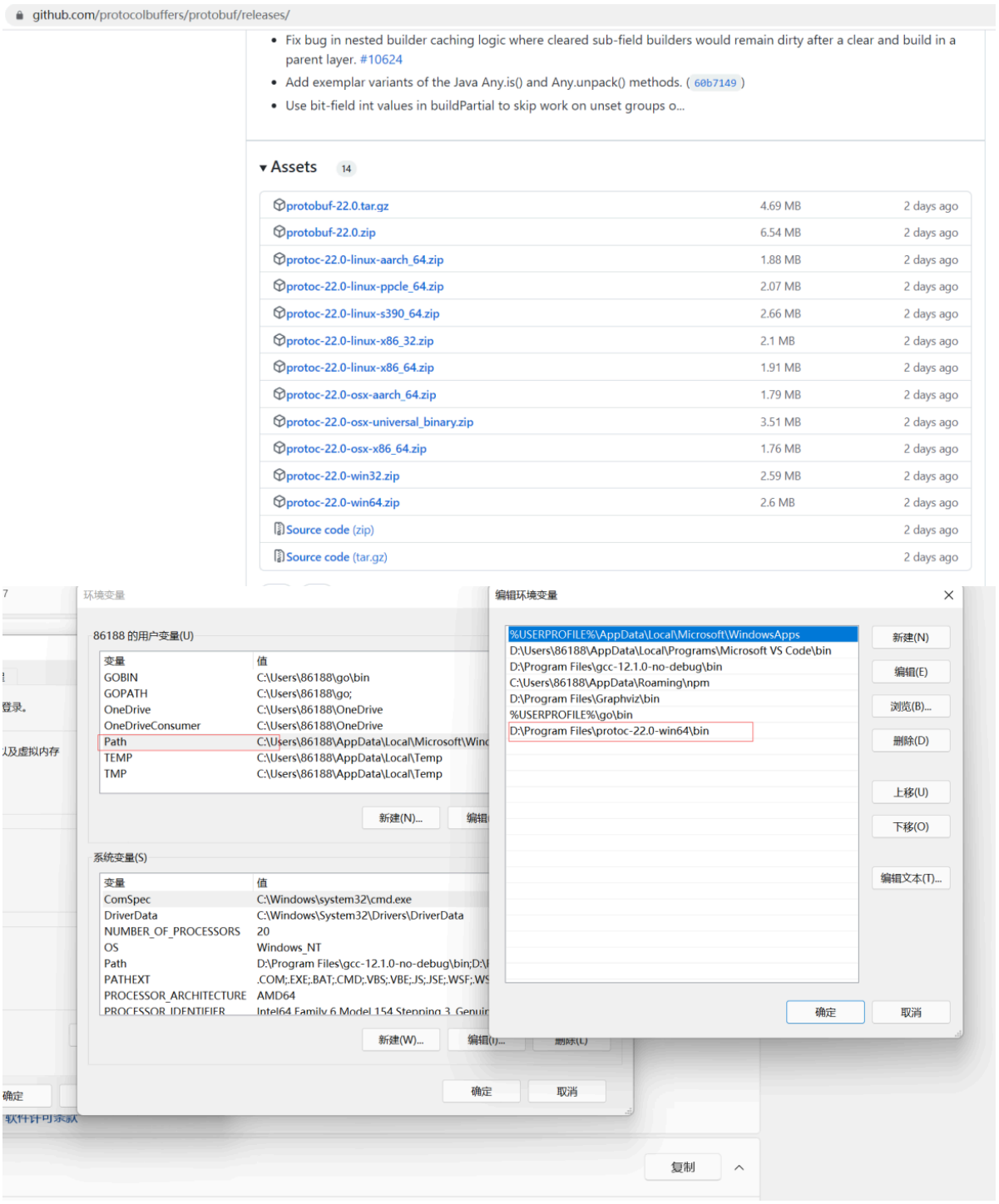
1. 分布式场景： gRPC 设计为低延迟和高吞吐量通信，非常适用于效率至关重要的轻型微服务
2. 点对点实时通信：gRPC对双向流媒体提供出色的支持，可以实时推送消息而无需轮询
3. 多语言混合开发：支持主流的开发语言，使gRPC成为多语言开发环境的理想选择
4. 网络流量受限环境：基于protobuffer序列化后的消息小的特征

protocol buffer

protoc

用于编译 .proto 文，生成对应语言的模板文件

```
# 下载地址
https://github.com/protocolbuffers/protobuf/releases/
```



protoc 插件安装

```
go install google.golang.org/protobuf/cmd/protoc-gen-go@v1.28
go install google.golang.org/grpc/cmd/protoc-gen-go-grpc@v1.2
```

protoc 编译

```
protoc --go_out=. --go_opt=paths=source_relative --go-grpc_out=. --go-
grpc_opt=paths=source_relative .\echo\echo.proto
```

1. --proto_path 或者 -I： 指定 import 路径，可以指定多个参数，编译时按顺序查找，不指定时默 认查找当前目录。 （.proto 文件中也可以引入其他 .proto 文件，这里主要用于指定被引入文件的 位置）
2. --go_out： golang编译支持，指定输出文件路径;

3. –-go_opt： 指定参数，比如 --go_opt=paths=source_relative 就是表明生成文件输出使用相对 路径。
4. path/to/file.proto： 被编译的 .proto 文件放在最后面

oauth2

```
go get -u golang.org/x/oauth2
```

健康检查

grpc健康检查探针

1. 源码：https://github.com/grpc-ecosystem/grpc-health-probe