

gRPC: Remote Procedure Call

Introduction to Middleware

Vojtěch Horký Petr Tůma

Department of Distributed and Dependable Systems
Faculty of Mathematics and Physics
Charles University

2010 – 2022

Outline

- 1 Technology Overview
- 2 Assignment Part I
- 3 Server Implementation
- 4 Client Implementation
- 5 Assignment Part II

Technology Overview

Goals

Provide platform independent remote procedure call mechanism.

Features

- Protocol buffers as interface description language.
- Stub code generation for multiple languages (C++, Java, Python, Go, Ruby, JavaScript, PHP, C# ...).
- Binary transport format with compact data representation.
- Supports streaming arguments during remote call.
- Synchronous and asynchronous invocation code.
- Compression support at transport level.
- Security support at transport level.

Examples To Begin With ...

```
> git clone http://github.com/d-iii-s/teaching-introduction-middleware.git
```

C

```
> cd teaching-introduction-middleware/src/grpc-basic-server/c  
> cat README.md
```

Java

```
> cd teaching-introduction-middleware/src/grpc-basic-server/java  
> cat README.md
```

Python

```
> cd teaching-introduction-middleware/src/grpc-basic-server/python  
> cat README.md
```

Service Specification Example

```
syntax = "proto3";  
  
message AnExampleRequest { ... }  
message AnExampleResponse { ... }  
  
service AnExampleService {  
  
    rpc OneToOneCall (AnExampleRequest) returns (AnExampleResponse) { }  
  
    rpc OneToStreamCall (AnExampleRequest)  
        returns (stream AnExampleResponse) { }  
  
    rpc StreamToStreamCall (stream AnExampleRequest)  
        returns (stream AnExampleResponse) { }  
  
}
```

Outline

- 1 Technology Overview
- 2 Assignment Part I**
- 3 Server Implementation
- 4 Client Implementation
- 5 Assignment Part II

Assignment

Server

Implement a server that will provide information on current time.

- The server should accept a spec of what fields to return.
- Fields should be standard YYYY-MM-DD HH:MM:SS.

Client

Implement a client that will query server time:

- Pick a random combination of fields.
- Query information on current time.
- Print the time.

Interoperability

Implement compatible clients and servers in two languages.

Outline

- 1 Technology Overview
- 2 Assignment Part I
- 3 Server Implementation**
- 4 Client Implementation
- 5 Assignment Part II

C++ Service Basics

Implementation

```
class MyService : public AnExampleService::Service {
    grpc.Status OneToOne (grpc.ServerContext *context,
        const AnExampleRequest *request, AnExampleResponse *response) {
        // Method implementation goes here ...
        return (grpc.Status::OK);
    }
    ...
}
```

Execution

```
MyService service;
grpc.ServerBuilder builder;
builder.AddListeningPort ("localhost:8888", grpc.InsecureServerCredentials ());
builder.RegisterService (&service);
std::unique_ptr<grpc.Server> server (builder.BuildAndStart ());

server->Wait ();
```

Java Service Basics

Implementation

```
class MyService extends AnExampleServiceGrpc.AnExampleServiceImplBase {  
    @Override public void OneToOne (  
        AnExampleRequest request,  
        io.grpc.stub.StreamObserver<AnExampleResponse> responseObserver) {  
        // Method implementation goes here ...  
        responseObserver.onNext (response);  
        responseObserver.onCompleted ();  
    }  
    ...  
}
```

Execution

```
io.grpc.Server server = io.grpc.ServerBuilder  
    .forPort (8888).addService (new MyService ()).build ().start ();  
  
server.awaitTermination ();
```

Python Service Basics

Implementation

```
class MyServicer (AnExampleServiceServicer):  
    def OneToOne (self, request, context):  
        # Method implementation goes here ...  
        return response
```

Execution

```
server = grpc.server (  
    futures.ThreadPoolExecutor (  
        max_workers = SERVER_THREAD_COUNT))  
add_AnExampleServiceServicer_to_server (MyServicer (), server)  
server.add_insecure_port ("localhost:8888")  
server.start ()
```

Outline

- 1 Technology Overview
- 2 Assignment Part I
- 3 Server Implementation
- 4 Client Implementation**
- 5 Assignment Part II

C++ Client Basics

Connection

```
std::shared_ptr<grpc.Channel> channel = grpc.CreateChannel (  
    "localhost:8888", grpc.InsecureChannelCredentials ());
```

Invocation

```
grpc.ClientContext context;  
AnExampleResponse response;  
std::shared_ptr<AnExampleService::Stub> stub = AnExampleService::NewStub (channel);  
grpc.Status status = stub->OneToOne (&context, request, &response);  
if (status.ok ()) {  
    // Response available here ...  
}
```

Java Client Basics

Connection

```
io.grpc.ManagedChannel channel = io.grpc.ManagedChannelBuilder
    .forAddress ("localhost", 8888)
    .usePlaintext ()
    .build ();
```

Invocation

```
AnExampleServiceGrpc.AnExampleServiceBlockingStub stub =
    AnExampleServiceGrpc.newBlockingStub (channel);
AnExampleResponse response = stub.oneToOne (request);
// Response available here ...
```

Python Client Basics

Connection

```
with grpc.insecure_channel ("localhost:8888") as channel:
```

Invocation

```
stub = AnExampleServiceStub (channel)
response = stub.OneToOne (request)
# Response available here ...
```

Show Your Code ...

Query Host Name

```
> hostname  
u1-22
```

Run Screen Sharing

```
> x11vnc -viewonly
```


Outline

- 1 Technology Overview
- 2 Assignment Part I
- 3 Server Implementation
- 4 Client Implementation
- 5 Assignment Part II**

Assignment

Performance

Measure the performance of your implementation.

Experiment Design

Stick to the following, or provide arguments for why not:

- Random field mix, each field with probability 1/2.
- Measure at least two minutes long traffic.
- Report average invocation throughput.
- No printing during measurement.
- Compare with past assignments.

Submission

GitLab

Use your personal GitLab repository under <https://gitlab.mff.cuni.cz/teaching/nswi163/2022>.

Requirements

- Use the assignment subdirectory.
- Write brief report in SOLUTION.md.
- Include build scripts with instructions.
- Do not commit binaries or temporary build artifacts.
- Tag your solution with task-03 and push the tag.