Eva Stepanek

MidEng 7.2 gRPC Framework GK

Dokumentation

Questions

1. **What is gRPC and why does it work across languages and platforms?**
   gRPC is a modern, open-source framework for remote procedure calls (RPC). It uses HTTP/2 for fast communication, Protocol Buffers (Protobuf) for efficient serialization, and supports multiple programming languages. This combination ensures seamless communication between systems on different platforms and in different languages.
2. **Describe the RPC life cycle starting with the RPC client.**
   The client creates a request with the required data and sends it to the server using a gRPC stub. The server receives the request, processes it, and returns a response. During this process, gRPC handles serialization, transport over HTTP/2, and deserialization, ensuring smooth communication.
3. **Describe the workflow of Protocol Buffers.**
   First, a .proto file is written to define structured data using specific syntax. The Protocol Buffers compiler (protoc) then generates code in the target programming language. This code is used to serialize data into compact binary format for transmission and deserialize it back into structured objects on the other side.
4. **What are the benefits of using protocol buffers?**
   - Compact binary serialization makes data smaller and faster to transmit compared to JSON or XML.
   - Cross-language support allows seamless integration across systems.
   - Backward compatibility ensures old and new versions of data can coexist.
   - It is optimized for performance, saving CPU and memory.
5. **When is the use of protocol not recommended?**
   Protocol Buffers are not ideal when:
   - Human-readable formats like JSON or YAML are needed (e.g., configuration files).
   - The project is small and doesn't require highly efficient serialization.
   - Dynamic or schema-less data is preferred since Protobuf requires a predefined schema.

6. **List 3 different data types that can be used with protocol buffers.**
    - int32 or int64 (for integers of different sizes).
    - string (for text data).
    - repeated (for arrays or lists).

Step-by-step

1. Hello World: https://grpc.io/docs/languages/java/quickstart/

```
PS C:\Users\evast> git clone -b v1.68.1 --depth 1 https://github.com/grpc/grpc-java
Cloning into 'grpc-java'...
remote: Enumerating objects: 3277, done.
remote: Counting objects: 100% (3277/3277), done.
remote: Compressing objects: 100% (2245/2245), done.
remote: Total 3277 (delta 651), reused 1789 (delta 238), pack-reused 0 (from 0)
Receiving objects: 100% (3277/3277), 3.68 MiB | 6.15 MiB/s, done.

Resolving deltas: 100% (651/651), done.
Note: switching to '16f93c8127ce60e3ee7d1276f60a5e6369ff5d7d'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

Updating files: 100% (2108/2108), done.
PS C:\Users\evast> cd grpc-java/examples
```

```
PS C:\Users\evast\grpc-java\examples> ./gradlew installDist
Downloading https://services.gradle.org/distributions/gradle-8.8-bin.zip
.....
.....

Welcome to Gradle 8.8!

Here are the highlights of this release:
 - Running Gradle on Java 22
 - Configurable Gradle daemon JVM
 - Improved IDE performance for large projects

For more details see https://docs.gradle.org/8.8/release-notes.html

Starting a Gradle Daemon (subsequent builds will be faster)

> Task :compileJava
Warnung: [options] Quellwert 8 ist veraltet und wird in einem zuk├╝nftigen Release entfernt
Warnung: [options] Zielwert 8 ist veraltet und wird in einem zuk├╝nftigen Release entfernt
Warnung: [options] Verwenden Sie -Xlint:-options, um Warnungen zu veralteten Optionen zu unterdr├╝cken.
Hinweis: Einige Eingabedateien verwenden oder ├╝berschreiben eine veraltete API.
Hinweis: Wiederholen Sie die Kompilierung mit -Xlint:deprecation, um Details zu erhalten.
Hinweis: C:\Users\evast\grpc-java\examples\src\main\java\io\grpc\examples\customloadbalance\CustomLoadBalanceClient.java
 verwendet nicht gepr├╝fte oder unsichere Vorg├ñnge.
Hinweis: Wiederholen Sie die Kompilierung mit -Xlint:unchecked, um Details zu erhalten.
3 Warnungen

BUILD SUCCESSFUL in 1m 19s
40 actionable tasks: 40 executed
PS C:\Users\evast\grpc-java\examples> ./build/install/examples/bin/hello-world-server
Nov. 26, 2024 8:35:26 AM io.grpc.examples.helloworld.HelloWorldServer start
INFORMATION: Server started, listening on 50051
```

Eva Stepanek

```
PS C:\Users\evast> cd grpc-java/examples
PS C:\Users\evast\grpc-java\examples> ./build/install/examples/bin/hello-world-client
Nov. 26, 2024 8:38:09 AM io.grpc.examples.helloworld.HelloWorldClient greet
INFORMATION: Will try to greet world ...
Nov. 26, 2024 8:38:09 AM io.grpc.examples.helloworld.HelloWorldClient greet
INFORMATION: Greeting: Hello world
PS C:\Users\evast\grpc-java\examples>
```