Introduction to gRPC using Scala

Zack Angelo
Software Architect, BigCommerce

@zackangelo

Zack Angelo

@zackangelo
zack.angelo@bigcommerce.com

- Software Architect @ BigCommerce
- Scala developer for 5 years and some change
- Leading the transition to gRPC at BigCommerce
- Send me an email/tweet, feedback welcome :)

Agenda

- gRPC Overview
- Rationale
- Getting started with Scala
- Tips/Tricks/Gotchas

What is gRPC?

- Service communication framework from Google and others
  - Based on four generations and 10 years of development at Google
- An alternative to building REST or HTTP+JSON API
- Automatically generated clients and server interfaces
- Broad language support
  - C++, Java, Go, Ruby, Node.js, Python, ObjC, PHP

Based on HTTP/2

- Per-stream flow control
  - Each request stream has it’s own flow control window
- Stream multiplexing
  - Many requests/streams, single connection
  - Everything is a stream
- Binary framing and transport
- Header compression
- Bidirectional streaming

vs Thrift and others

- Superior backwards compatibility characteristics
  - All fields are optional
  - Enums are forwards and backwards
- Doesn’t attempt to hide the network
  - Async stubs

IDL describes your API

- Service define methods
- Each method has a request and a response message type
- Each message type has a field with a corresponding type, name and tag number

```
syntax = "proto3";

service FortuneCookie {
    rpc NextFortunes(NextFortuneReq)
        returns (NextFortuneResp);
}

message NextFortuneReq {
    //number of fortunes to return
    int32 num_fortunes = 1;
}

message NextFortuneResp {
    repeated string fortunes = 1;
}
```

Why gRPC?

(or, why did we choose it?)

Operability

- Uniform service transport allow us to create tools and techniques around building and running these services at scale
  - Tracing
  - Health Checking
  - Monitoring/Metrics
- Canonical service naming and discovery

Type Safety on the Wire

- In most cases will prevent dynamically typed languages from sending one type as another (e.g., string “1234” for an `int32` field)
- Many dynamic client libraries will throw an exception at runtime if you attempt to set a field with the wrong type
- A common problem for us when interacting with PHP
  - Especially when consuming from Scala

Code Generation

- BigCommerce has a polyglot stack:
  - PHP
  - Scala
  - Java
  - Node.js
  - Ruby
  - Go
- Great code generation for C++/Java/Go, decent for others
  - Canonical JSON representation in Proto3
- Async stubs where possible, doesn’t hide the network
- Were being crushed by the weight of having to maintain client code for each service, even with very few services
- Want each service to have implicit access to another

Performance

- Persistent, multiplexed connections
  - Reduced TLS overhead
- No URL/query string parsing
- Binary serialization
  - Most CPU bound tasks in our services relate to serialization and deserialization
- Google: 3X increase in throughput, 75% reduction in CPU on Cloud PubSub
  - https://cloud.google.com/blog/big-data/2016/03/announcing-grpc-alpha-for-google-cloud-pubsub

Getting Started

Language Support

Want to learn more?

Get started by learning concepts and doing our hello world quickstart in language of your choice.

GET STARTED

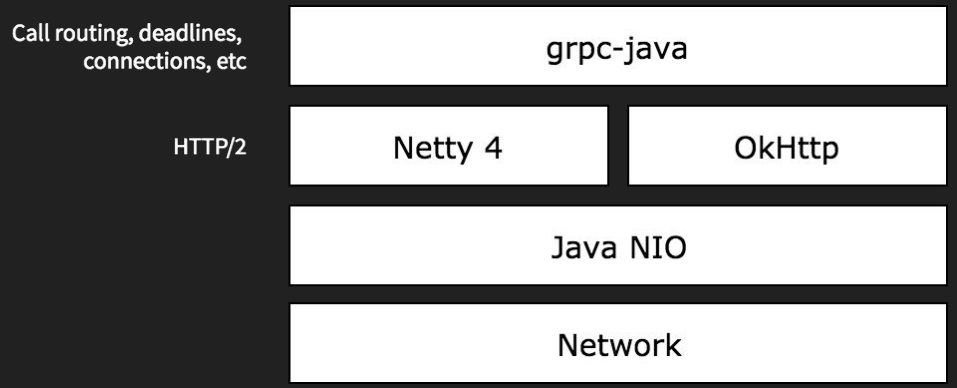
Or go straight to Quick Start in the language of your choice:

???

C++	Java	Python
Go	Ruby	C#
Node.js	Android Java	Objective-C
	PHP	

grpc-java Architecture

- Non-blocking
- Pluggable HTTP/2 transport
  - Netty 4
  - OkHttp (client only)



Use grpc-java directly?

- Generates immutable Java classes + builders
- Uses Java collections
  - Requires `scala.collection.JavaConverters`
- Uses Guava Futures / `StreamObserver` interfaces
  - Not straightforward to use Scala futures
- No thanks.

ScalaPB

This repository Search Pull requests Issues Marketplace Gist

scalapb / ScalaPB

Watch 36 Unstar 401 Fork 67

Code Issues 15 Pull requests 2 Projects 0 Wiki Insights

Protocol buffer compiler for Scala. https://scalapb.github.io/

scalapb protocol-buffers scala

679 commits 12 branches 76 releases 21 contributors Apache-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download

thesamet Proptest: more restrictions on generated enum labels ... Latest commit 268318c 12 days ago

.github	Add ISSUE_TEMPLATE.md	a year ago
compiler-plugin	Don't generate invalid code on empty packages	12 days ago
e2e	Don't generate invalid code on empty packages	12 days ago

ScalaPB

- gRPC and Protobuf 3 support for Scala
- Built on top of the gRPC Java stack
- Proto messages are modeled using case classes, `Option`, and Scala collections
- gRPC stubs use Scala futures
- SBT integration

```
syntax = "proto3";

service FortuneCookie {
    rpc NextFortunes(NextFortuneReq)
        returns (NextFortuneResp);
}

message NextFortuneReq {
    //number of fortunes to return
    int32 num_fortunes = 1;
}

message NextFortuneResp {
    repeated string fortunes = 1;
}
```

```
trait FortuneCookies extends AbstractService {
  override def serviceCompanion = FortuneCookies
  def nextFortunes(request: NextFortuneReq): Future[NextFortuneResp]
}
```

```
final case class NextFortuneReq(numFortunes: Int = 0)
  extends GeneratedMessage
  with Message[NextFortuneReq]
  with Updatable[NextFortuneReq]
```

Proto to Scala Field Types

Protobuf	**Scala**	**Default Value**
int32/int64	Int/Long	0
double/float	Double/Float	0.0
string	String	“”
message	Option[T]	None
repeated	Seq[T]	Seq.empty[T]
enum	sealed trait	Zero tag value

Demo

Tips/Tricks/Gotchas

`Option[_]` Fields

- All message fields are set to `None` when omitted
- All primitive fields are set to a default value when omitted
- Use Google's "well known types" to signal to ScalaPB that you'd like a field modeled as an `Option`.

```
import "google/protobuf/wrappers.proto";

message MyMessage {
    google.protobuf.StringValue optional_string = 1;
}
```

```
case class MyMessage(optionalString: Option[String] = None)
```

Error Handling

- Errors must be sent out of band or defined in every response message
- You can return built-in codes and simple messages by throwing a StatusException
- For more elaborate cases, create a protobuf message to represent your error payload
- Use the `trailers` constructor arg to pass your error payload over the wire
- For more detailed exception to error mapping, use an interceptor

```
message FortuneError {
    string code = 1;
    string message = 2;
}
```

```
val errorMarshaller = new Metadata.BinaryMarshaller[FortuneError] {
  override def toBytes(value: FortuneError): Array[Byte] =
    value.toByteArray
  override def parseBytes(serialized: Array[Byte]): FortuneError =
    FortuneError.parseFrom(serialized)
}

val errorKey = Metadata.Key.of("error-bin", errorMarshaller)
```

```
val metadata = new Metadata()
metadata.put(errorKey, FortuneError("invalid_id", "Invalid ID specified"))

//adds an error trailer in response
throw new StatusException(Status.FAILED_PRECONDITION, metadata)
```

Context Propagation

- A gRPC Context is a generic container for state that should cross thread boundaries
  - E.g. credentials, tracing
- Facilitates cancellation
- Available for use in a separate artifact: `io.grpc % grpc-context % 1.3.0`
- Solution: Use a custom, delegating ExecutionContext that overrides `prepare()` to capture and transfer state from call site to callback
  - Similar to techniques used for logging context propagation (e.g., MDC in logback)

Direct Executor

- If your service implementation is non-blocking, use `.directExecutor()` to bypass the transport's thread pool
- Eliminates a possible bottleneck and reduces request latency
- If you do accidentally block, you'll probably ruin Christmas for someone

```
val server = NettyServerBuilder
  .forAddress(new InetSocketAddress(interface, port))
  .addService(FortuneCookiesGrpc.bindService(Impl, EC))
  .directExecutor()
  .build()
```

PATCH Semantics

- How can I instruct an RPC method to only update certain fields?
- JSON can express three field states: populated, null and absent
- Protobuf is unable to easily detect the absence or presence of a field
- Option 1: Use a fetch then CAS update
- Option 2: Use a protobuf “field mask” (breaks compile-time safety, requires you to pass field names in strings)

Load Balancing

- Client-side
  - Linkerd (http://linkerd.io)
  - Envoy (https://lyft.github.io/envoy)
- Traditional
  - HAProxy (TCP)
- Google Cloud Global LB
  - Alpha coming in Q3

Thanks!

Q&A