Adopting gRPC at Spotify

@mattgruter
2019-02-06
“Developers don’t care about new RPC technologies”

-- someone at KubeCon 2018
Adopting gRPC at Spotify

@mattgruter
2019-02-06
Why?

Why are we doing this?

What?

What do we get out of this?

How?

How do we get there?
Spotify’s Infrastructure

~2500 services
~1000 developers
~250 teams
Java, Python, ...
Hermes

Not this Hermes!
Hermes

But this
Hermes

Written in 2012

Based on ZeroMQ

JSON or protobuf payload

Not a RPC framework
Hermes works!
Why Change?
Hermes Ecosystem
404 Not Found
From NIH to OSS
Why gRPC?
Projects

We host and nurture components of cloud native software stacks, including Kubernetes, Prometheus, and Envoy. Kubernetes and other CNCF projects are some of the highest velocity projects in the history of open source. We are regularly adding new projects to better support a full stack cloud native environment.

Kubernetes is the world’s most popular container-orchestration platform and the first CNCF project. Kubernetes helps users build, scale and manage modern applications and their dynamic lifecycles. First developed at Google, Kubernetes now counts more than 2,300 contributors and is used by some of the world’s most innovative companies, across a wide range of industries. The cluster scheduler capability lets developers build cloud native applications, while focusing on code rather than ops. Kubernetes future-proofs application development and infrastructure management on-premises or in the cloud, without vendor or cloud-provider lock-in.
gRPC

1. HTTP/2 based
2. Binary protocol
3. Strongly typed service and message definition (Protobuf)
4. Encryption
The gRPC Advantage
Code Generation

Java, Golang, Python, Ruby, Dart, PHP, Node.js, Objective-C, C#, C++
syntax = "proto3";

package spotify.metadata.v1;

option java_package = "com.spotify.metadata.v1";
option java_multiple_files = true;
option java_outer_classname = "MetadataProto";

// Interface exported by the server.
service Metadata {
  rpc GetMetadata(SongId) returns (SongMetadata) {} 
}

message SongId {
  int32 id = 1;
}

message SongMetadata {
  int32 id = 1;
  string name = 2;
  string artist = 3;
  string album = 4;
}
public class MetadataService extends MetadataGrpc.MetadataImplBase {

    // [...] 

    @Override
    public void getMetadata(SongId songId,
            StreamObserver<SongMetadata> response) {
        LOG.info("Received getMetadata request");
        response.onNext(store.searchMetadata(songId)
                .orElse(EMPTY_METADATA));
        response.onCompleted();
    }
}
func main() {
    ctx, cancel := context.WithTimeout(
        context.Background(), time.Second)
    defer cancel()

    conn, err := grpc.Dial("nls://metadata")
    if err != nil {
        log.Fatalf("couldn't connect to grpc server: %v", err)
    }
    defer conn.Close()

    client := pb.NewMetadataServiceClient(conn)
    r, err := client.Metadata(ctx,
        &pb.SongId{
            Id: "42"
        },
    )
    if err != nil {
        log.Printf("metadata request failed: %v\n", err)
        return
    }
    fmt.Println(r.Response)
}

Polyglot client impl.
@Override
public void getMetadata(SongId songId,
LOG.info("songId: " + songId);
response.onNext(
    orElseGet(
        MetadataGrpc.MetadataImplBase
    
Description
copied from
class:
Fetch metadata for a given song.
A feature with an empty name is returned if there's no feature at the given position.

Overrides:
getMetadata in class MetadataImplBase

metadata.main
Schema Management

1. Embrace the proto

2. Shared repo for all protos

3. Version on the proto level
Schema Management
github.com/uber/prototool
Resiliency
Deadlines

Ingress v2 -> metadata-proxy (2000 ms)

Spotify

metadata-proxy -> metadata-storage (1500 ms)
A common RPC
Retries

- Transparent
- Configurable
Hedging

Basic Hedging Pathway

Client Application \rightarrow gRPC Client Lib \rightarrow Multiple Backends

- Client makes gRPC Call
- Sends RPC
- Returns OK
- Sends RPC to each backend
- Fastest backend sends OK
- Sends cancel to other backends

Client receives response and status
Retry throttling
Load Balancing & Routing
Load Balancing

- Client-side
- Proxy
- Lookaside
Load Balancing

lookaside
How to Migrate?
Our Journey
Stats

2874 Services
1341 HTTP
983 Hermes
76 gRPC
Our Journey

80 services
Distance Travelled

Hermes

900 services
Remaining Distance to Destination

gRPC
Challenge #1

Change is hard
1. Don’t force it!
1. Don’t force it!

code generation
1. Don’t force it!

Resiliency patterns

code generation
1. Don’t force it!

- Tracing
- Resiliency patterns
- code generation
1. Don’t force it!

- Tracing
- Code generation
- Resiliency patterns
- Istio
2. Make the Right Choice
the Easy Choice
3. Unblock & Decouple
$ grpcurl -max-time 1 -plaintext -d '{"requests": {"kind": 1, "gid": "[REDACTED]"}}' metadataproxy [REDACTED] spotify.metadata.service.Metadata/GetEntities
{
    "responses": [
    {
        "entity": {
            "track": {
                "gid": [REDACTED],
                "name": "Never Gonna Give You Up",
                "album": {
                    "gid": [REDACTED],
                    "name": "The Best Of",
                    "artist": [
                        {
                            "gid": [REDACTED],
                            "name": "Rick Astley",
                            "localized_name": []
                        }
                    ]
                }
            }
        }
    }
}
Challenge #2

Yet another protocol
Never-ending migration
Step by step

1. Add a new gRPC API
2. Move clients to new API
3. Remove old API
Step by step

1. Add a new gRPC API
2. Move clients to new API
3. Remove old API

this is hard!
Challenge #3

Developer experience
Our Journey

80 services
Distance Travelled

900 services
Remaining Distance to Destination

Hermes

gRPC
Our Journey

Hermes

gRPC
Our Journey

Hermes  ->  gRPC
“Developers don’t care about new RPC technologies”

-- someone at KubeCon 2018
Developers don’t have care about new RPC technologies
Thank You