Google: gRPC

return to Google page

Table 1: Data Sheet for gRPC

Characteristic	Value
Original author(s)	Google
Developer(s)	Google
Initial release	February 2015[1)]
Stable release	1.21.1 https://packages.grpc.io/
Repository	https://github.com/grpc/grpc/releases
Written in	
Operating system	Linux, Windows 7+,Mac[2)]
Software Languages	C/C++, C#, Dart, Go, Java, Node.js, PHP, Python, Ruby[3)]
Available in	English
Type	
License	Apache 2.0 [4)]
Website	https://grpc.io/about/

Overview

Source: The following is from the about gRPC page

gRPC is a modern open source high performance RFC1831 - Remote Procedure Call Protocol Specification Version 2 (RPC) framework that can run in any environment. It can efficiently connect services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication. It is also applicable in last mile of distributed computing to connect devices, mobile applications and browsers to backend services.

The main usage scenarios:

- *Efficiently connecting polyglot services in microservices style architecture*
- *Connecting mobile devices, browser clients to backend services*
- *Generating efficient client libraries*

Core Features:

- *Idiomatic client libraries in 10 languages*
- *Highly efficient on wire and with a simple service definition framework*
- *Bi-directional streaming with http/2 based transport*
- *Pluggable auth, tracing, load balancing and health checking*

Architecture

Source: The following is from "Introduction to gRPC"

gRPC has two parts, the gRPC protocol, and the data serialization. By default gRPC utilizes Protobuf for serialization, but it is pluggable with any form of serialization you wish to use, with some caveats, which I will get to later.

The Protocol

The protocol itself is based on http2, and exploits many of its benefits. Google's development of SPDY laid down the first designs for what eventually became http2.

gRPC supports several built in features inherited from http2, such as compressing headers, persistent single TCP connections, cancellation and timeout contracts between client and server.

The protocol has built in flow control from http2 on data frames. This is very handy for insuring clients respect the throughput of your system, but does add an extra level of complexity when diagnosing issues in your infrastructure, because either client or server can set their own flow control values.

Load balancing (LB) is normally performed by the client, which chooses the server for a given request from a list provided by a Load Balancing server. The LB server will monitor the health of endpoints and use this and other factors to manage the list provided to clients. Clients will use a simple algorithm such as round-robin internally, but note that the LB server may apply more complex logic when compiling the list for a given client.

RPC Types

gRPC offers two essential types for client server communication.

Unary

Essentially these are synchronous requests made to the gRPC server with a single request that blocks until a response is received.

Streaming

Streaming is really powerful and can be accomplished in three different configurations: client pushing messages to a stream; server pushing messages to a stream; or bidirectional, where client and server are both sending data in two streams in the same method. In all cases the client initiates the RPC method.

Streams don't provide any acknowledgement of receipt until the stream completes, which can add

complexity when the system needs to cope with node failures or network partitions. This can be mitigated by using a bidirectional stream to return ACKs. If a server is given a chance to kill a connection gracefully a message will be returned indicating the last received message.

1)

Jason Smith, Container Solutions, Mar 1, 2017, https://container-solutions.com/introduction-to-grpc/

2) 3)

, The gRPC about page https://grpc.io/about/

4)

gRPC FAQ, https://grpc.io/faq/