gRPC as a Wire Protocol for Data Systems

Angelo Kastroulis
akastroulis@carrera.io
Carrera Group

ABSTRACT

This paper proposes a novel method for communicating with data systems over the wire using gRPC as a stable wire format over a purpose-built alternative using TCP sockets.

1 INTRODUCTION

Clients and servers must interact over a network. A variety of methods are used to accomplish the task in application servers (such as HTTP), however, data systems typically have a closed protocol that is accessible only through a client application (drivers, etc.) [7].

2 BACKGROUND AND RELATED WORK

2.1 Wire Protocols

The client and server components of a data system communicate over the network using *wire protocol*. PostgreSQL's protocol is called *PGWire* [7]. It defines specific interactions between clients and server (described in Figure 1).

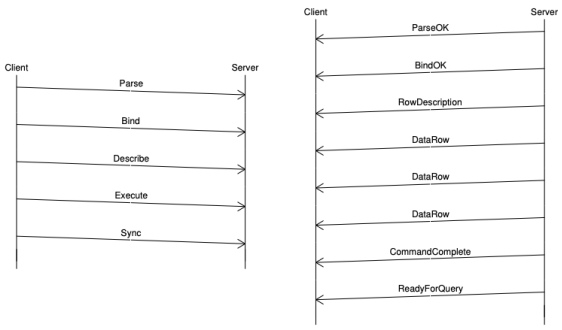

Figure 1: PGWire extended protocol interaction.

The protocol specifies that the first few bytes determine the command, followed by several flow control interactions. The server responds with an acknowledgement and streams rows, before issuing that it is ready to accept new commands.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.2 gRPC

gRPC is a modern, fast approach to making procedure calls across a network. While it isn't as fast as a pure socket implementation, it is significantly more performance than HTTP [2, 4, 8]. It has also seen effective use in technologies that require server-to-server interactions such as Tensorflow to synchronize work across training instances [3] and CockroachDB for partition replication [6]. PostgreSQL steering committees have also discussed the idea of supporting a gRPC-based wire protocol [1].

3 METHODOLOGY/DESIGN

3.1 Wire Protocol

I propose a method of using gRPC for node to node communication using *protobuf* as the wire protocol schema. Both the client and server are nodes. The client does not expose protobuf to the end user. Since serialization and deserialization are expensive operations [5]. To minimize the impact, the schema is as compactly as possible (see 1).

The schema and the payload are both bytes so that the wire protocol is not tightly coupled to the data structures. That allows you to load an Avro document, for example, into the payload and schema properties and send them along without the wire protocol being affected. Additionally, that allows you to completely avoid the cost of serialization (and deserialization) of the payload.

Listing 1: Record schema for insert RPC

```
message Record {
  required bytes schema = 1;
  required bytes payload = 2;
}
```

3.2 Client Abstraction

As mentioned in the previous section, the client would not expose the wire schema (or even the fact that protobuf exists) to the caller. The client is effectively a simple wrapper for the wire protocol as shown in Figure Figure 2).

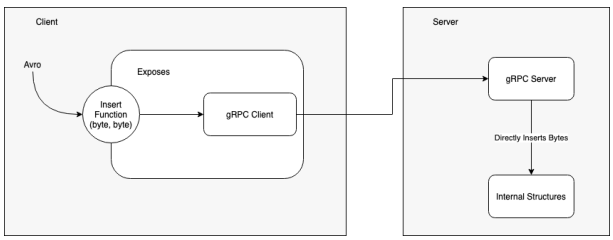

Figure 2: Client/Server interaction

4 EVALUATION

When new wire protocols for PostgreSQL are proposed, they first must benchmark the number of queries per second (*QPD*) can be pushed through the new protocol (because performance is key).

I propose a similar method of evaluation in comparison to HTTP. The test would be simple: build a simple gRPC procedure with the schema protobuf. Take a binary document (Avro, etc.) and measure the number of times per second it could push through the interface (it is not necessary to test the data system or other components). A simple function that acks simulating the server will suffice.

The comparison to the benchmark could be anything currently in use (an existing protocol, or HTTP). HTTP would similarly follow the same method by sending an Avro document across HTTP, and the server simply has an empty ack response without any further processing.

5 CONCLUSION

A simple, flexible gRPC wire protocol could do the job of replacing HTTP with very little effort. There is no need to code schemas for the actual data into gRPC protobuf, nor is there value in forcing a database to change to using protobuf.

REFERENCES

[1] PostgreSQL HTTP2 Wire Format. https://postgrespro.com/list/thread-id/2378911. (????). Accessed: 2021-05-12.

[2] RSocket vs. gRPC Benchmark. https://dzone.com/articles/rsocket-vs-grpc-benchmark. (????). Accessed: 2021-05-12.

[3] Rajarshi Biswas, Xiaoyi Lu, and Dhabaleswar K Panda. 2018. Designing a micro-benchmark suite to evaluate gRPC for TensorFlow: Early experiences. *arXiv preprint arXiv:1804.01138* (2018).

[4] Carolina Luiza Chamas, Daniel Cordeiro, and Marcelo Medeiros Eler. 2017. Comparing REST, SOAP, Socket and gRPC in computation offloading of mobile applications: An energy cost analysis. In *2017 IEEE 9th Latin-American Conference on Communications (LATINCOM)*. IEEE, 1–6.

[5] Henri Hagberg et al. 2019. Performance of serialized object processing in Java. (2019).

[6] Lakshmi Narasimhan Seshan, Rajesh Jalisatgi, and Vijaeendra Simha GA. ACID Compliant Distributed Key-Value Store. (????).

[7] Jan Urbaski. 2016. Postgres on the wire. (2016).

[8] Bairen Yi, Jiacheng Xia, Li Chen, and Kai Chen. 2017. Towards zero copy dataflows using rdma. In *Proceedings of the SIGCOMM Posters and Demos*. 28–30.

A APPENDI