CS6650 – Vancouver campus
Spring 2022

Evaluating gRPC performance in real time applications

Comparing gRPC against REST APIs in remote real time string manipulation tasks.

Francisco Proboste

Hyphotesis and research question

To objective of this experiment is to understand to what extent gRPC is a high performant protocol to connect distributed services in real time applications.

1. How does gRPC compares to Rest API in real time string manipulation tasks?
2. How is the distribution of individual-words tasks latency?

Tests

A simple remote task requesting invert words will be used to compare the performance of the two mechanisms : API Rest v/s gRPC.

"Speed" → "Deesp"

Experiment 1:

One by one word inversion.

- 300 words aprox.
- No pause between words.

Experiment 2:

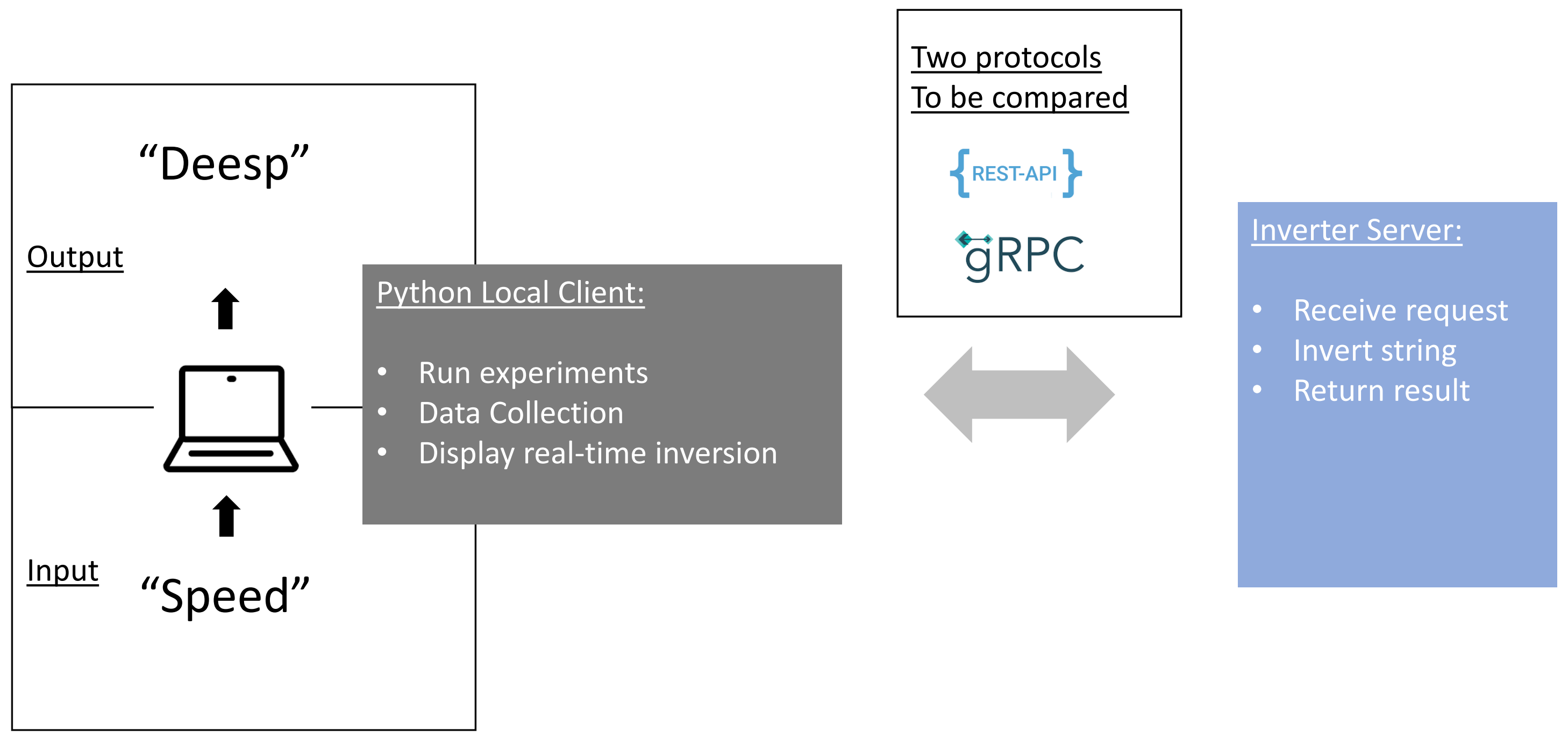
Large text inversion.

- 300 words aprox.
- Test behavior with a larger payload.

Experiment 3:

Experiment 1 in a local deployment to understand the differences without transport.

Architechture Diagram



"Deesp"

Output

Input

"Speed"

Python Local Client:

- Run experiments
- Data Collection
- Display real-time inversion

Two protocols
To be compared

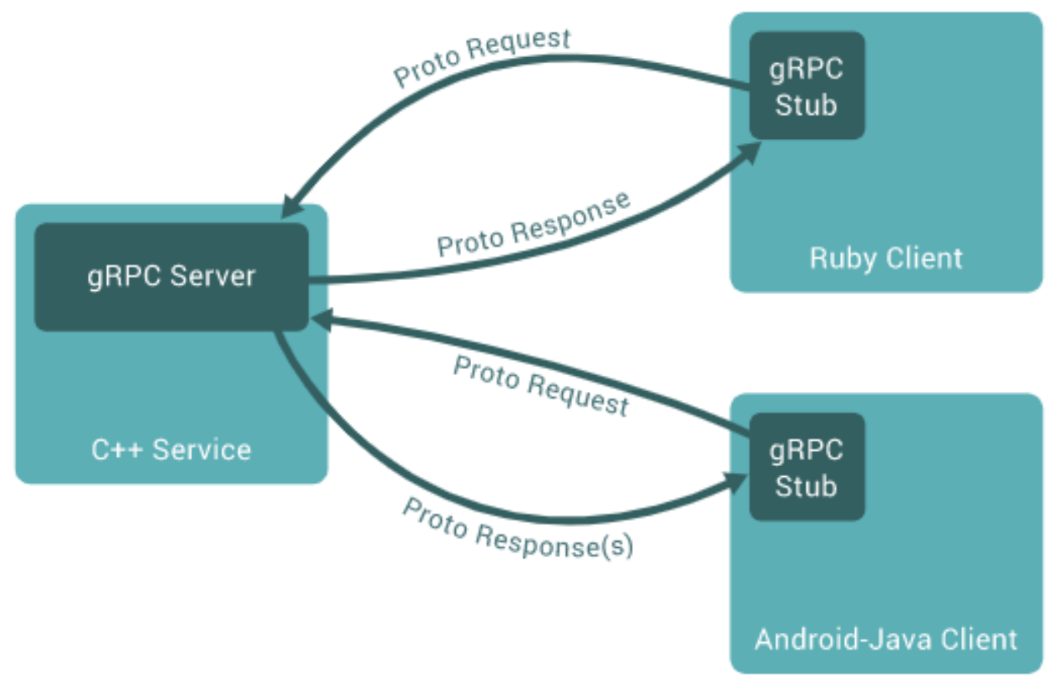
REST-API

gRPC

Inverter Server:

- Receive request
- Invert string
- Return result

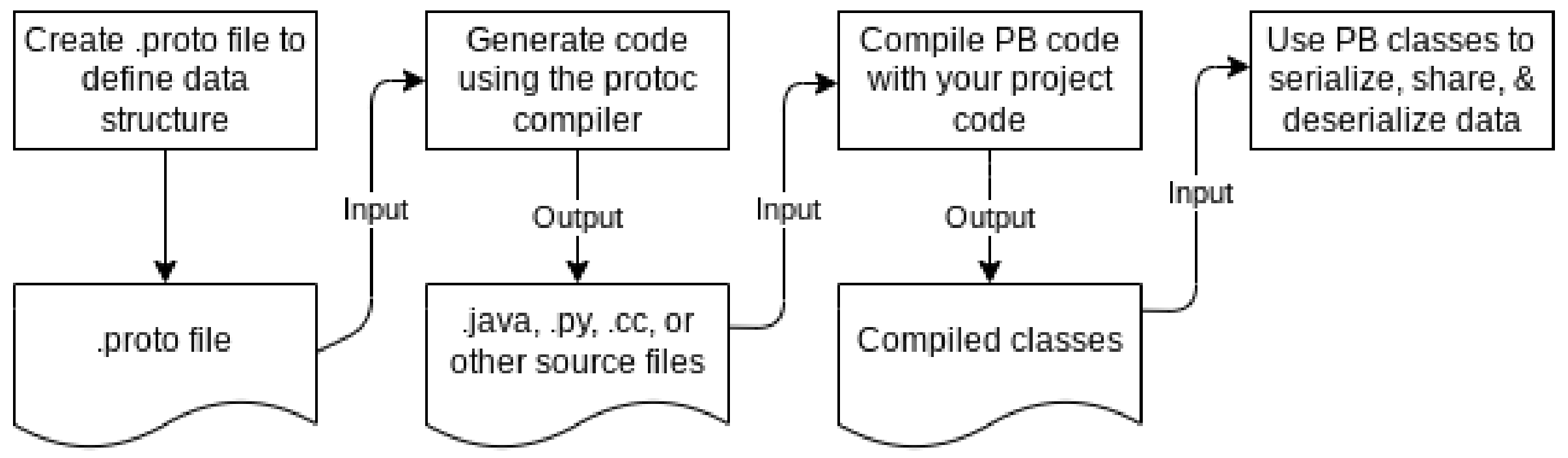
gRPC implementation (oficial example)



Source: https://grpc.io/docs/what-is-grpc/introduction/

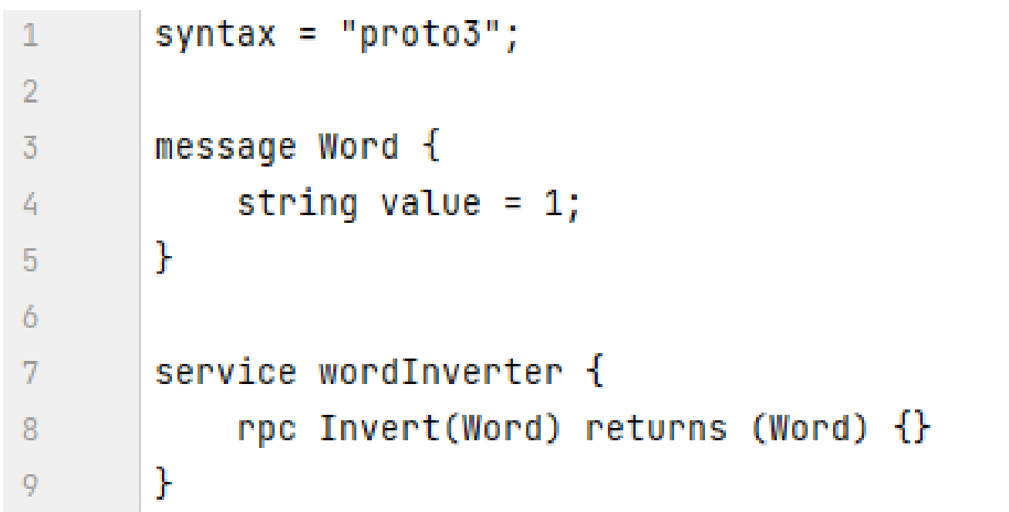
Protocol buffers

Multi-language, bidirectional data serialization mechanism.


Create .proto file to define data structure
.proto file
Input
Generate code using the protoc compiler
Output
.java, .py, .cc, or other source files
Input
Compile PB code with your project code
Output
Compiled classes
Input
Use PB classes to serialize, share, & deserialize data

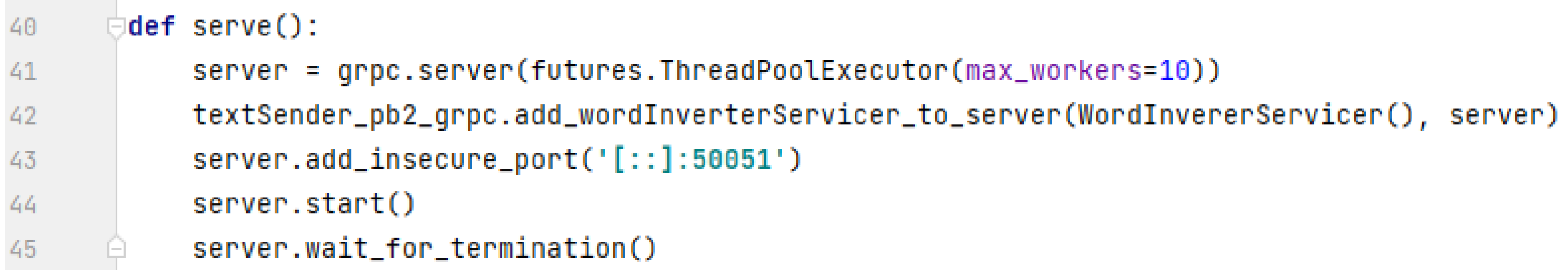
Source: https://developers.google.com/protocol-buffers/docs/overview

textSender.proto



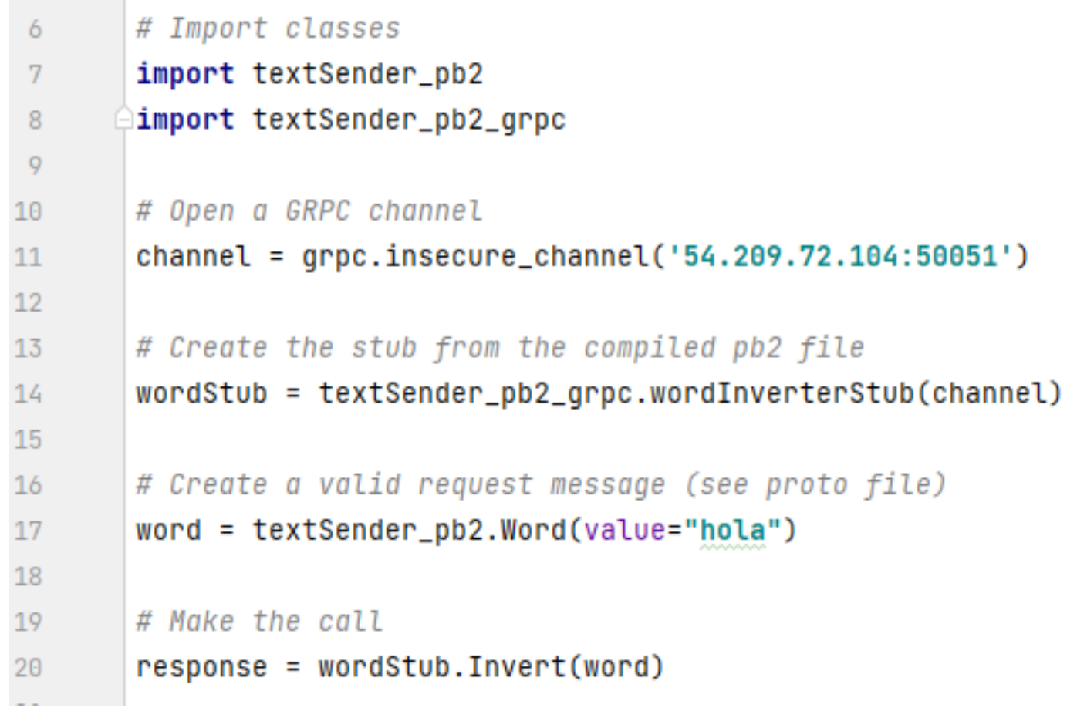
```
syntax = "proto3";

message Word {
    string value = 1;
}

service wordInverter {
    rpc Invert(Word) returns (Word) {}
}
```

Server instantiation



```
def serve():
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
    textSender_pb2_grpc.add_wordInverterServicer_to_server(WordInvererServicer(), server)
    server.add_insecure_port('[::]:50051')
    server.start()
    server.wait_for_termination()
```

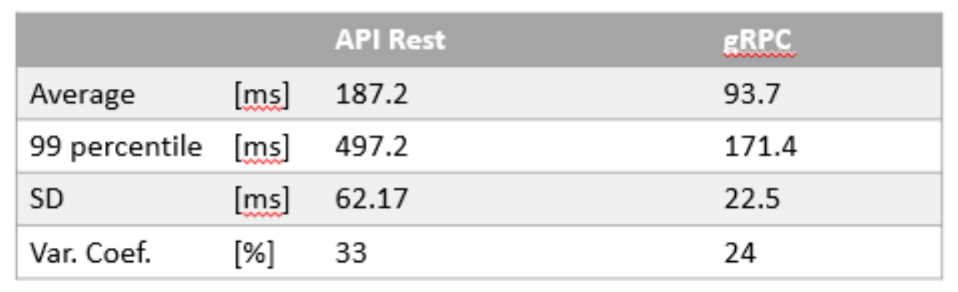
Client



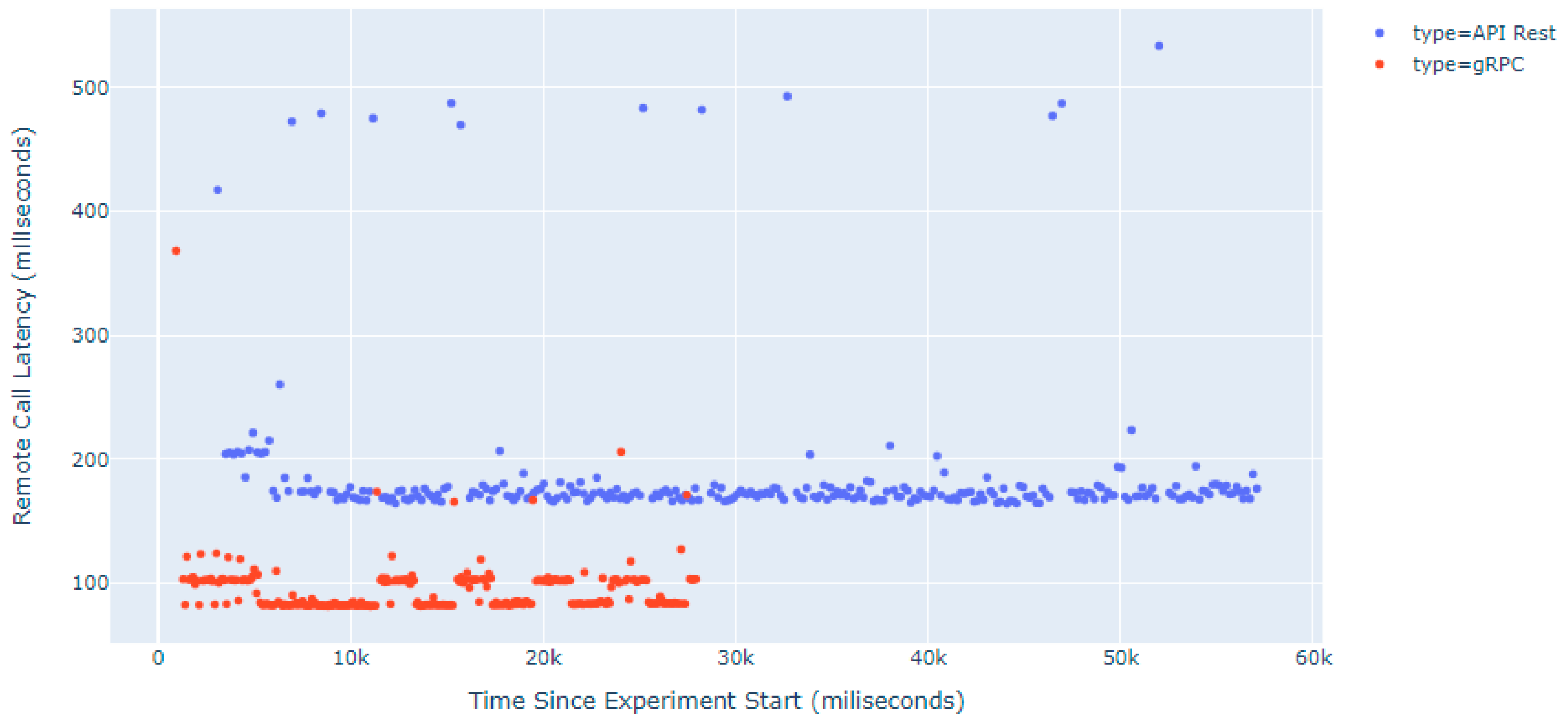
```
# Import classes
import textSender_pb2
import textSender_pb2_grpc

# Open a GRPC channel
channel = grpc.insecure_channel('54.209.72.104:50051')

# Create the stub from the compiled pb2 file
wordStub = textSender_pb2_grpc.wordInverterStub(channel)

# Create a valid request message (see proto file)
word = textSender_pb2.Word(value="hola")

# Make the call
response = wordStub.Invert(word)
```

Experiment 1

One by one inversion of 300 words aprox.

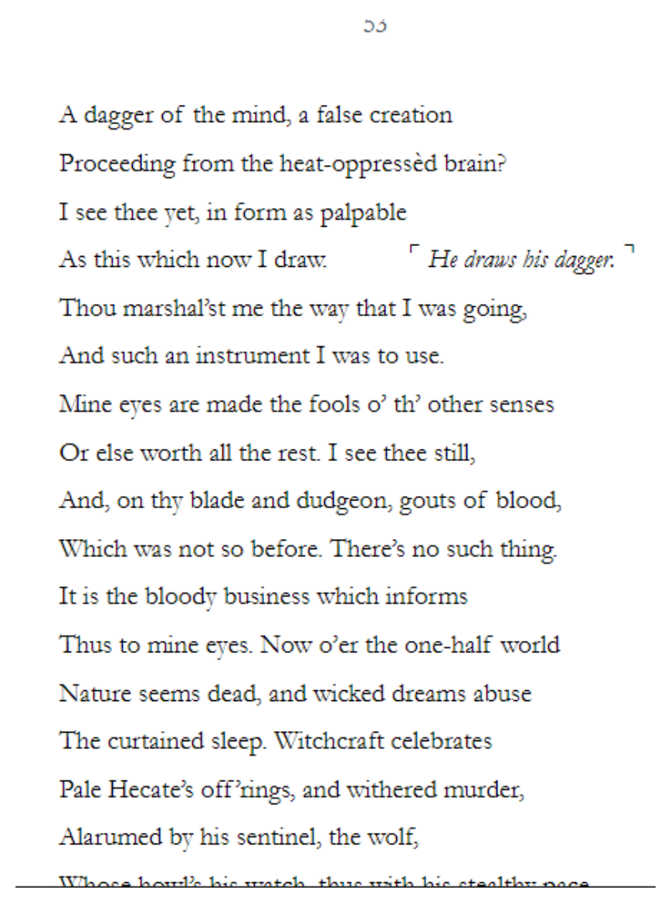


		API Rest	gRPC
Average	[ms]	187.2	93.7
99 percentile	[ms]	497.2	171.4
SD	[ms]	62.17	22.5
Var. Coef.	[%]	33	24



Experiment 2

Large payload inversion: 300 words in one call



55

A dagger of the mind, a false creation
Proceeding from the heat-oppressèd brain?
I see thee yet, in form as palpable
As this which now I draw. ⌜*He draws his dagger.*⌝
Thou marshal'st me the way that I was going,
And such an instrument I was to use.
Mine eyes are made the fools o' th' other senses
Or else worth all the rest. I see thee still,
And, on thy blade and dudgeon, gouts of blood,
Which was not so before. There's no such thing.
It is the bloody business which informs
Thus to mine eyes. Now o'er the one-half world
Nature seems dead, and wicked dreams abuse
The curtained sleep. Witchcraft celebrates
Pale Hecate's off'rings, and withered murder,
Alarumed by his sentinel, the wolf,
Whose howl's his watch, thus with his stealthy pace

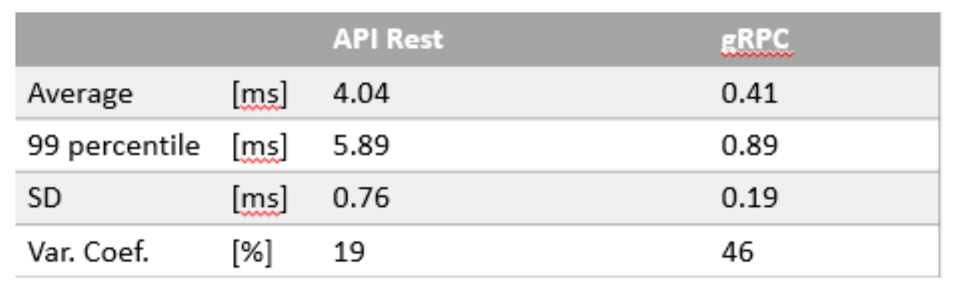
One Remote Call
To invert the whole text.

API Rest: 195.5 ms

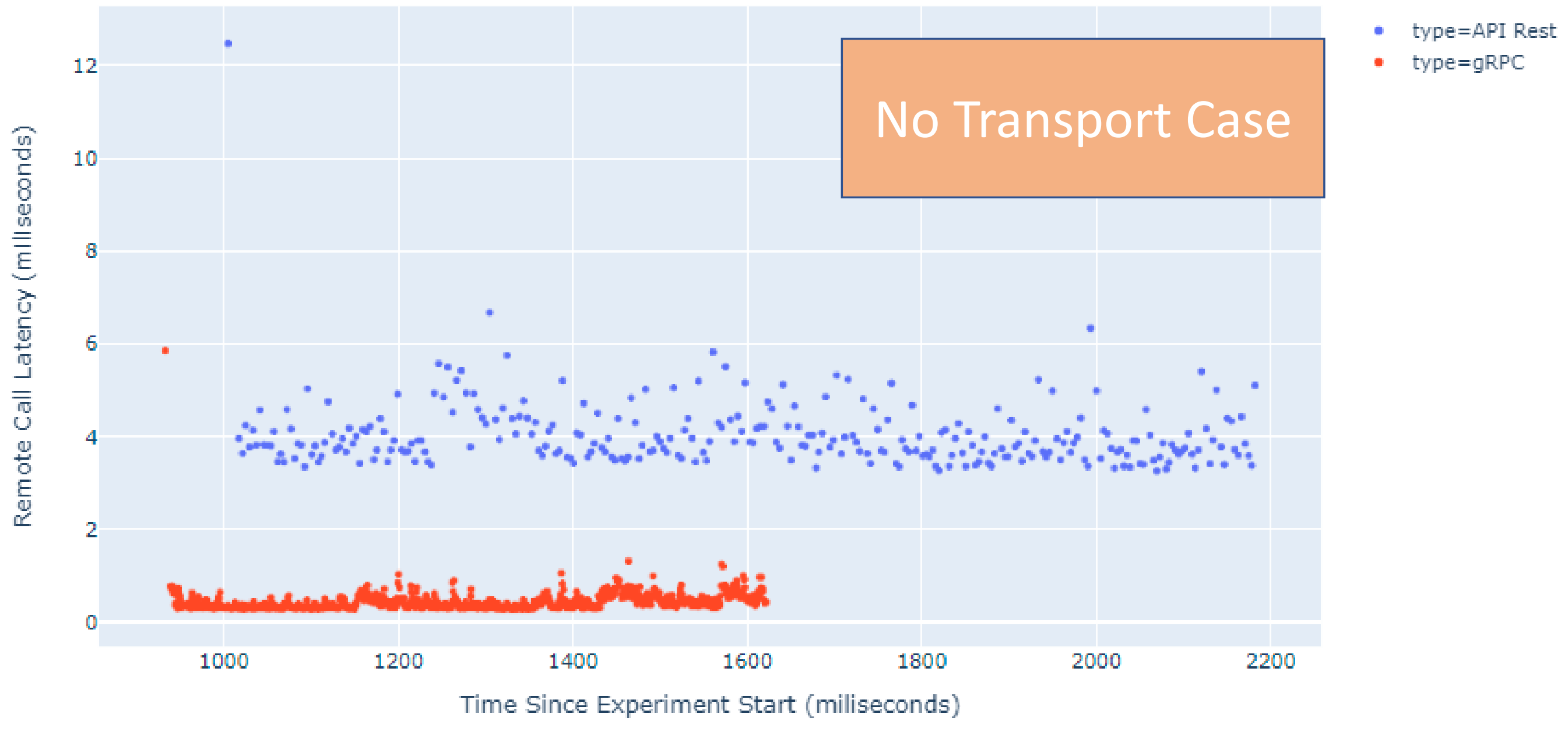
gRPC: 101.1 ms

Experiment 3

One by one inversion of 300 words aprox.



		API Rest	gRPC
Average	[ms]	4.04	0.41
99 percentile	[ms]	5.89	0.89
SD	[ms]	0.76	0.19
Var. Coef.	[%]	19	46



Conclusions

- gRPC is at least two times faster than API Rest in a simple low size load manipulation task on the cloud.
- gRPC could be up to one order of magnitude faster in short distance applications.
- Difficult to conclude which shows more variability in speeds.

Future questions

- How do they compare when servers are near capacity?
- How do they compare in term of parallel requests handling?
- How do they compare in resources requirements? Which is more suitable for IoT communication?