Distributed Software Architecture Lab 2 Report

# GOALS

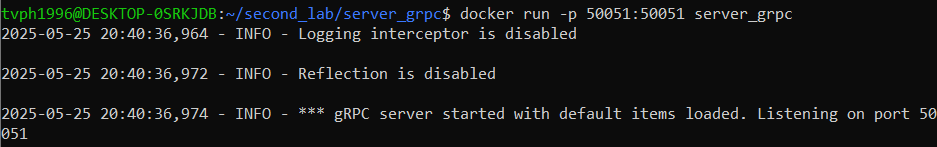
The purpose of this project is to get used to gRPC and analyze the difference in performance between gRPC and REST.

# github

<https://github.com/tvph1996/second_lab>

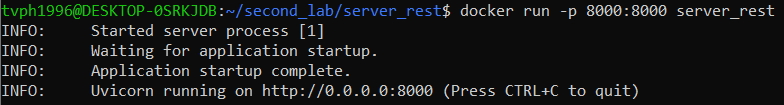
# output

#### gRPC-server starts



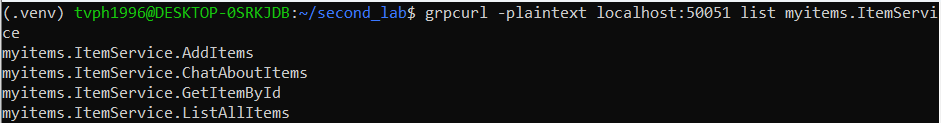
Port 50521

#### REST-server starts



Port 8000

#### gRPC-server Reflector

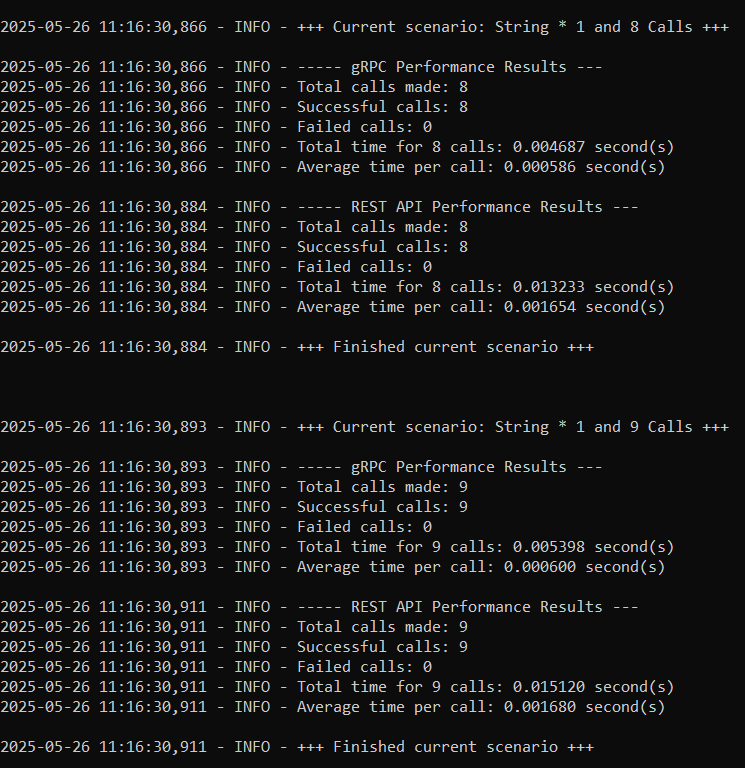


#### gRPC-server Interceptor Logging

Latency is under 1ms

#### gRPC-client interacting with gRPC-server

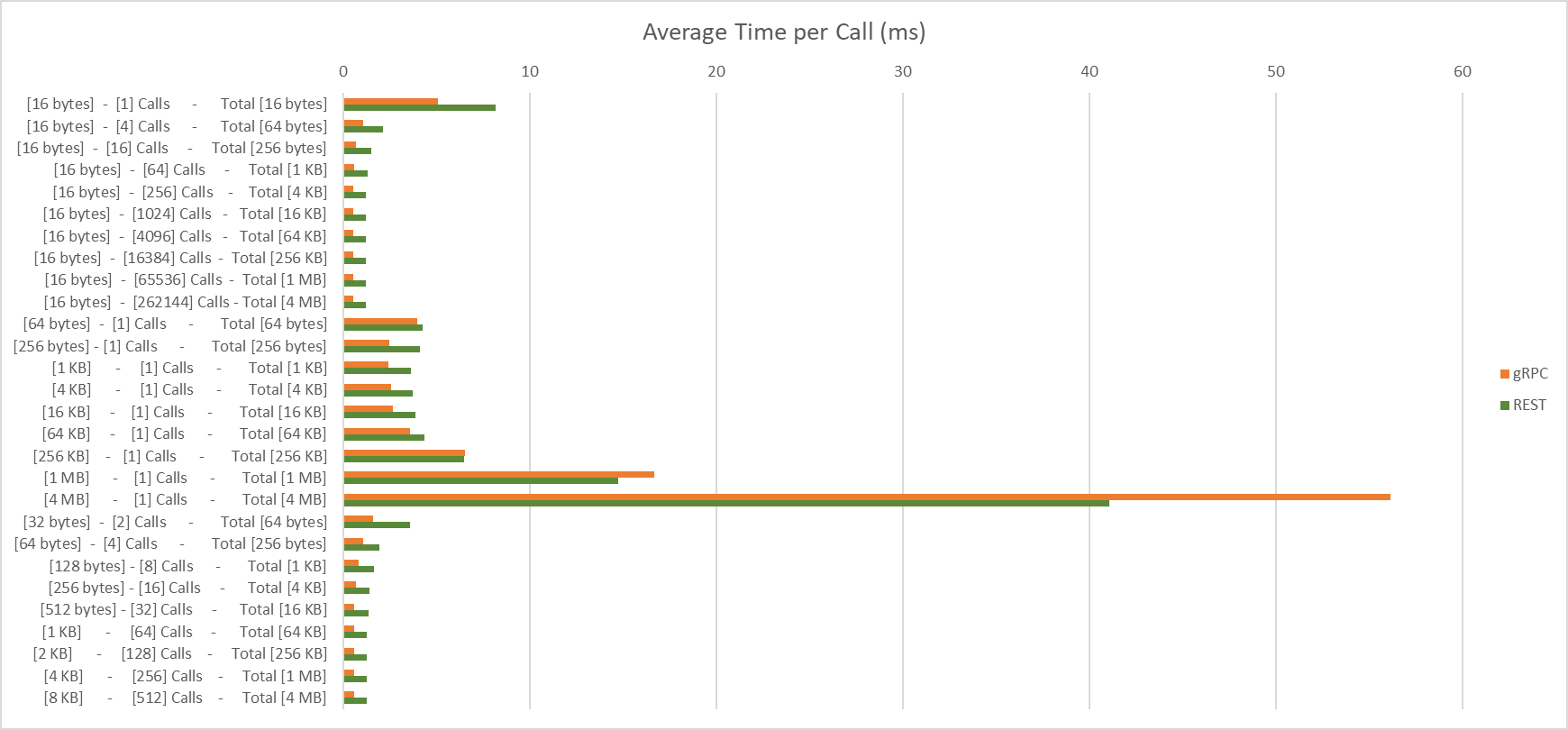
#### Performance Test Output

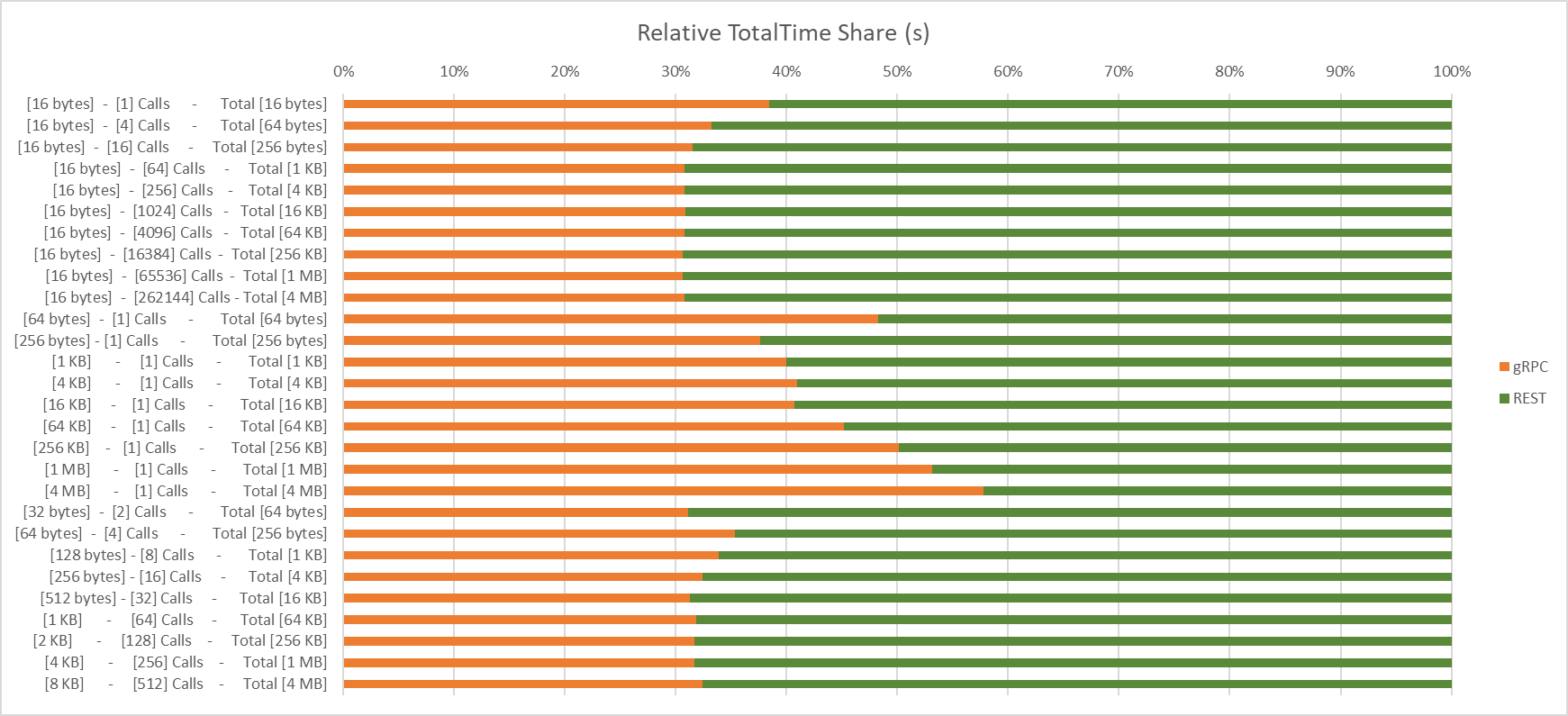


# PERFORMANCE TEST

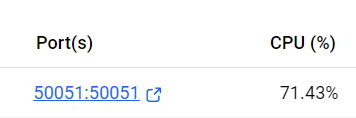
* To make the comparison more precise:
* REST-server is rebuilt using FastAPI instead of Flask to reduce overhead.
* Make sure the logic and the measurement is the same.
* **Overview**
* Method used for test: GetItemById
  + Request message: ItemId
  + Response message: ItemId & ItemName (string)
* Measurement:
  + Starts before established connection
  + Ends after received OK response from server
* From the starting point with the 16 bytes message:
  + Increase the number of calls by 4 times each
  + Increase the message length by 4 times each, keep the number of calls
  + Increase both the message length and the number of calls by 2 times
* The idea is to maintain the same size of message
* Run each scenario 10 times and taking the average
* **Results**

Detailed data is in github.

* **Charts**



* **Key insights**



* + gRPC utilize CPU much more than REST (~75% vs ~60%)
  + On Average, gRPC performs around 2 times better than REST
  + Interestingly, gRPC starts to perform worse than REST in case of heavy message (more than 256kb).
    - Actually, gRPC blocks message more than 4 MB, for the purpose of testing, special configuration must be added.
  + To transfer big data in gRPC, it’s good to find ideal combination between size of message and number of calls
    - Average Time per Call is low if message < 4 KB (based on data, can not conclude).
    - Number of Calls should be high enough to flat the waste of overhead when connection is established.
    - For example: for 4 MB message, it is much faster to call 512 times of 8 KB message compared to 262144 calls of 16 bytes message
  + Here, gRPC can achieve more than 100 mbps (4MB in 0.3224177s). This should be more than enough for:
    - 1080p streaming: ~8mbps and 4K streaming: ~16mbps
    - Compare to real world:
* gRPC server in real world is much faster since the server in this lab is running in Docker in a laptop
* Latency in real world is higher via internet connection, for this lab everything is local