Grand Challenge -1

CMPE 275

Hardik ghori (016452059)

Abraham Mathew (016018990)

Varun raj Badri (015918006)

Introduction

Implementing a low-level Socket based multi- threaded server client communication model
Using various technologies like grpc,Protobuf,Java and Python
System Requirements-Macos

Protobuf

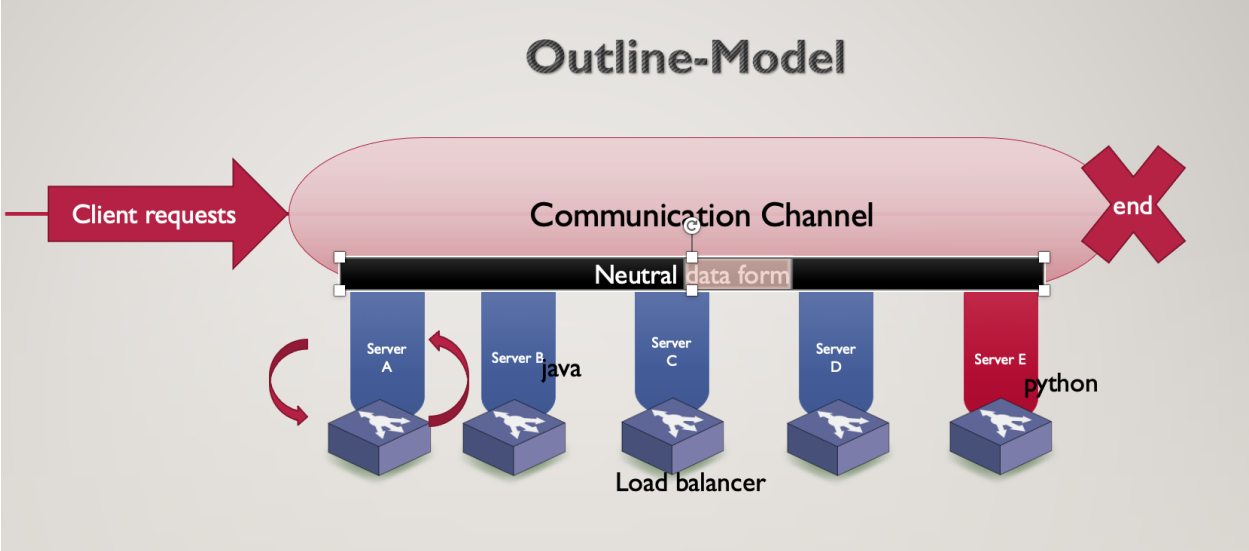
Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data – think XML, but smaller, faster, and simpler. You define how you want your data to be structured once, then you can use special generated source code to easily write and read your structured data to and from a variety of data streams and using a variety of languages.

gRPC

gRPC, a client application can directly call a method on a server application on a different machine as if it were a local object, making it easier for you to create distributed applications and services. As in most RPC systems, gRPC works by defining a service, specifying the methods that can be called remotely, along with their parameters and return types. On the server side, the server implements this interface and runs a gRPC server to handle client calls. On the client side, the client has a stub

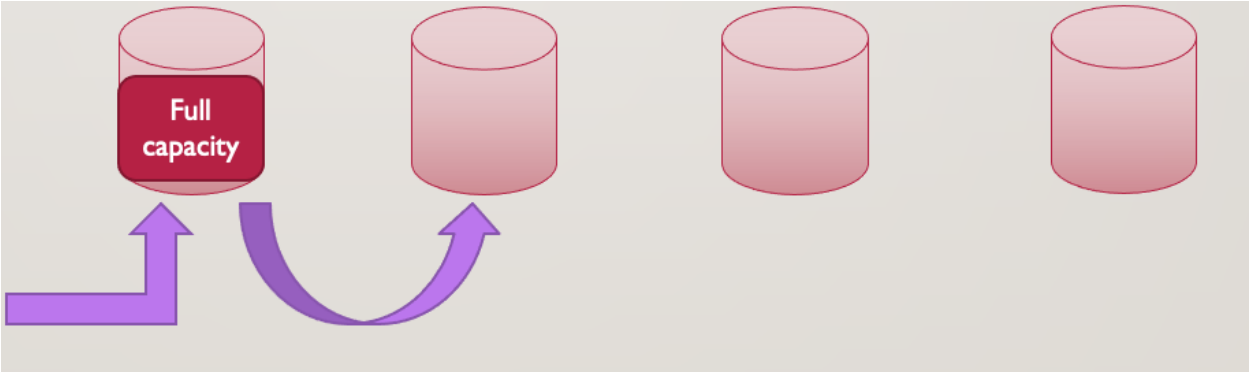
(referred to as just a client in some languages) that provides the same methods as the server.

Interact with gRPC and protobuf-based server-to-server communication
When a threshold is reached, the server can start the next server, and a load balancer will control each server instance for a particular service.
In the event that the current server's capacity is reached, a Python module will construct servers dynamically. If the server cannot connect to the following server, it will attempt to connect to the following server. Send requests with a String payload. Through java and python client libraries implementation on the server side between various load balancers

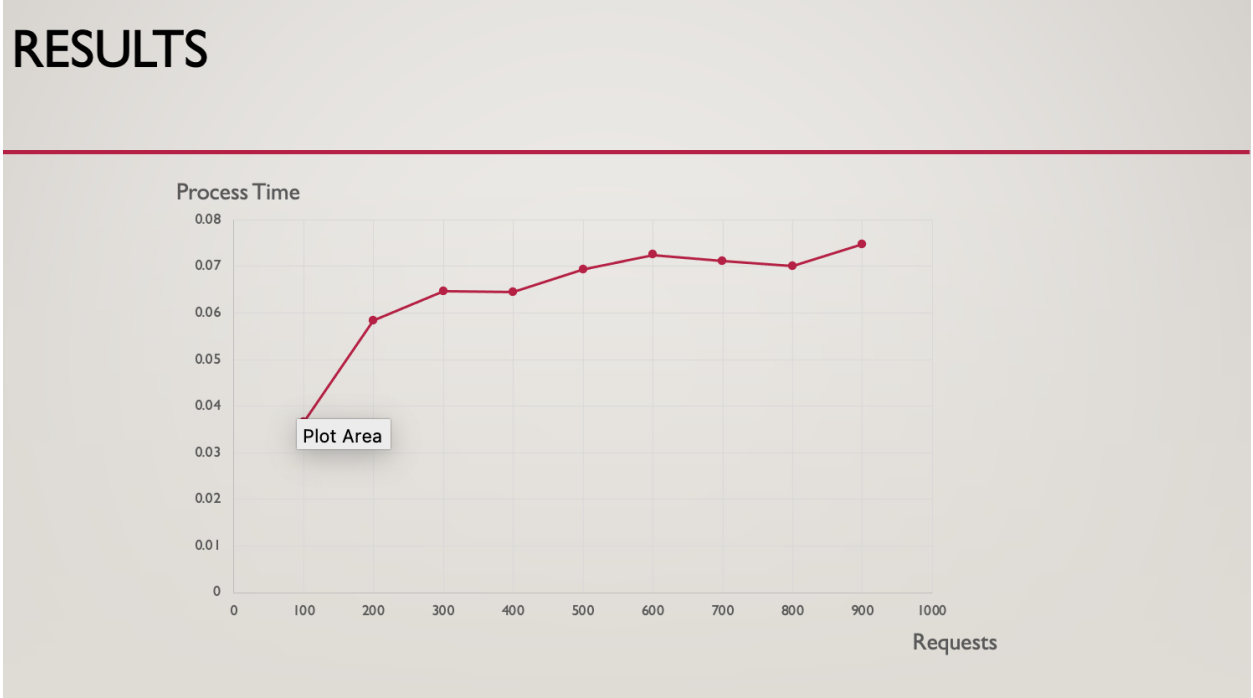


•Unlike centralized architecture because of failure(leader), we took a 50-50 approach.

•Traffic distribution is in a dynamic manner by passing it to the next server in case if storage is not available. And the same goes on.



The time number of requests took to process using this design is as follows:



pythonServerMessage_pb2.py > ...

```
2   # Generated by the protocol buffer compiler.  DO NOT EDIT!
3   # source: pythonServerMessage.proto
4   """Generated protocol buffer code."""
5   from google.protobuf import descriptor as _descriptor
6   from google.protobuf import descriptor_pool as _descriptor_pool
7   from google.protobuf import message as _message
8   from google.protobuf import reflection as _reflection
9   from google.protobuf import symbol_database as _symbol_database
10  # @@protoc_insertion_point(imports)
11
12  _sym_db = _symbol_database.Default()
13
14
15
16
17  DESCRIPTOR = _descriptor_pool.Default().AddSerializedFile(b'\n\x19pythonServerMessage.proto
18
19
20
21  _MESSAGE = DESCRIPTOR.message_types_by_name['Message']
22  _MESSAGERESPONSE = DESCRIPTOR.message_types_by_name['MessageResponse']
23  Message = _reflection.GeneratedProtocolMessageType('Message', (_message.Message,), {
```

Output

```
server 4 checking 2
---> putting 7 w destination 2
server 2 checking 2
server 0, thread 0 doing work 15
server 2 checking 2
server 4 checking 2
server 4 checking 2
---> putting 8 w destination 3
server 0, thread 0 doing work 17
server 4 checking 2
server 4 checking 2
---> putting 9 w destination 4
server 4 checking 2
server 0, thread 0 doing work 19
server 4 checking 2
```

```
server 4 checking 2
---> putting 8 w destination 3
server 0, thread 0 doing work 17
server 4 checking 2
server 4 checking 2
---> putting 9 w destination 4
server 4 checking 2
server 0, thread 0 doing work 19
server 4 checking 2
--> Put added 10 work requests
----------------------------
Server 0, sum: 5, w: 2
Server 1, sum: 7, w: 2
Server 2, sum: 9, w: 2
Server 3, sum: 11, w: 2
Server 4, sum: 13, w: 2
----------------------------
--> terminating test
Clock shutting down
```

Difficulties

- Lots of possibilities to execute but difficult to analyze them in a quick manner
- Condition where some servers fail and other all servers are full, this will fail
- Selection of leader in a dynamic way

Future Scope

- Load Balancer should be able to run services based on the number of request that it is receiving
- Leader Election