CSE-531_Project-1 ---gRPC Project

Yun Shing Lu

1. Project Statement

For this project, I am going to implement an interface using gRPC to build a distributed banking system so the customers as the client side and the branches as the server side can communicate with each other. By doing this framework, customers can withdraw, deposit, and query money from multiple branches in the bank.

we also assume that there are no concurrent updates on the bank account. Each branch maintains a replica of the money that needs to be consistent with the replicas in other branches. The customer communicates with only a specific branch that has the same unique ID as the customer. Although each customer independently updates a specific replica, the replicas stored in each branch need to reflect all the updates made by the customer.

2. Project goal

The goal for this project is we are going to pass the JSON file which contains the multiple events for customers and the account balance for each branch. Our banking system will follow the instructions from the input JSON file and return the corresponding output txt file.

3. Setup

The relevant technologies I used are as below:

Development environment:

- Python: 3.8.8
- IDE: VScode
- anaconda: 2021.05

Python packages:

- Multiprocessing
- concurrent
- grpc: 1.32.0

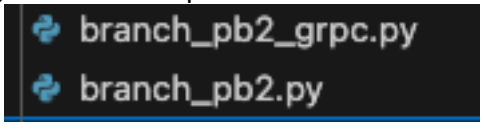
4. Implementation Processes

Step1: Define the serializing structured data in the .proto file.

It a combination of the definition language (created in .proto files), the code that the proto compiler generates to interface with data, language-specific runtime libraries, and the serialization format for data that is written to a file.

Step2: Execute the .proto file and generate .py source files

In this step, it will be generated two pb files as below and we don't need to edit them.



Step3: Create Branch.py as server side

In the Branch.py file, we need to define the branch behavior and build the sever

1. Create stub: setup gRPC channel & client stub for each branch
2. Message delivery: get the incoming requests from customer transaction and branch propagation
3. Process message: handle the received message (query, deposit, and withdraw) and return the response message
4. Propagate message: propagate the response message to branches

Step4: Create Customer.py as client side

In the Customer.py file, we need to define the customer behavior and build the client

1. Create stub: setup gRPC channel & client stub for branch
2. Execute events: send gRPC request for each event
3. Output message: generate output message

Step5: Create Main.py to control the program

1. Start branch gRPC server process
2. Start customer gRPC client processes
3. Parse JSON file and create objects
4. Output the result txt file

5. Results

Accoding to the program, it will generate the txt output file corresponding to the input JSON file. We can see the JSON format below including the id, interface, result, balance.

{'id': 1, 'recv': [{'interface': 'deposit', 'result': 'success'}, {'interface': 'query', 'balance': 410}]}
{'id': 1, 'recv': [{'interface': 'withdraw', 'result': 'success'}, {'interface': 'query', 'balance': 890}]}