

## CSE-531\_Project-3 ---Client-Centric 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.

Unlike Project 1, where the customer communicates with only a specific branch with the same unique ID, Project 3 allows the customer to communicate with different branches for query, withdrawal, and deposit operations.

### 2. Project goal

- Extend the implementation of Branch.Withdraw and Branch.Deposit interface to enforce Monotonic Writes policy.
- Extend the implementation of Branch.Withdraw and Branch.Deposit interface to enforce Ready your Writes policy.

### 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

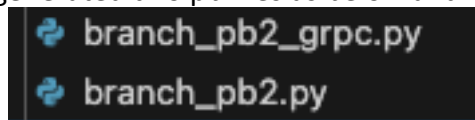
### 4. Implementation Processes

**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.



```
branch_pb2_grpc.py
branch_pb2.py
```

**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. Update Write set: generate new event ID, append to write set
4. Process message: handle the received message (query, deposit, and withdraw) and return the response message
5. Propagate Transaction: propagate the Customer transactions to other 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 JSON file

### **5. Results**

According to the program, it will generate two JSON output files corresponding to the monotonic-writes and read-your-writes JSON input files. We can see the JSON format below including the id, interface, result, balance.

```
[{"id": 1, "balance": 0 }]
```

```
[{"id": 1, "balance": 400 }]
```