CSE 531: Distributed and Multiprocessor Operating Systems

Client-Centric Consistency Written Report

Problem Statement

The problem statement involves implementing the read-your-writes client-centric consistency model on top of the distributed banking system that we had built for Project 1. The system has multiple branches and customers, and the challenge is to ensure read-your-writes consistency among the branch processes for a given customer. This consistency model requires that the effect of a write operation by a customer on a data item (e.g.,balance in our case) should always be visible in subsequent read operations by the same customer across different branches.

Goal

The goal is to implement the read-your-writes consistency model for a distributed bank system. Specifically, the project aims to track read and write events by the same customer across different branch processes and ensure that the effect of a write operation is consistently seen in successive read operations by the same customer, even when the customer switches between branches. This is accomplished by the following objectives:

- Define a service in a .proto file.
- Generate server and client code / stubs using the protocol buffer compiler present in the grpcio-tools package in python
- Use the Python gRPC API to create client and server components.
- Implement communication between customer and branch processes and branch and branch processes using gRPC.
- Implement interfaces (Query, Deposit, and Withdraw) for customer-branch and branch-branch interactions.
- Ensure that the system follows read-your-writes client centric consistency (from the customers perspective) by blocking further reads until writes are propagated among branches.
- Write the events received by the customer to the output file as mentioned in the sample output file.

This ensures that even if a customer switches between branches after making an update to their balance, the customer observes a consistent view of their balance.

Setup

Installations:

- Install Python 3.9 or above (I used Python 3.9).
- Install pip version 9.0.1 or higher.
- (OPTIONAL) If necessary, upgrade your version of pip:
  ```
  $ python -m pip install --upgrade pip
  ```
  If you cannot upgrade pip due to a system-owned installation, you can run the example in a virtualenv
  ```
  $ python -m pip install virtualenv
  $ virtualenv venv
  $ source venv/bin/activate
  $ python -m pip install --upgrade pip
  ```
- Install gRPC:
  ```
  $ python -m pip install grpcio
  ```
- Install gRPC tools:
  ```
  $ python -m pip install grpcio-tools
  ```

Running the project:

- Unzip from canvas
- Navigate to the folder where main.py is present.
- This step is OPTIONAL:
  If you want to regenerate the code generated by the gRPC library run:
  ```
  $ python -m grpc_tools.protoc -I./protos --python_out=. --pyi_out=.
  --grpc_python_out=. ./protos/banking_system.proto
  ```
  This will regenerate the files: banking_system_pb2.py, banking_system_pb2.pyi, banking_system_pb2_grpc.py
- One main requirement is that if your input file has branches from 1 to n then there should be n empty ports starting from 50001 to run the gRPC branch servers. E.g. If the branch id is 9 then its server runs on port 50009. If the ports are not available on your machine then you can change the value of BASE_PORT in constants.py to a value that suits your machine.
- Run:
  ```
  $ python main.py --input_file "./inputs/input_big.json"
  ```
  where --input_file is followed by the path of your input file.
- The output will be present in ./outputs/output.json (The path is relative to the folder containing main.py)
- In order to verify the output using the checker file you can run:
  ```
  $ python "./checkers/checker.py" "./outputs/output.json"
  ```

Implementation Processes

The implementation processes involve:

- Defining a service in a .proto file for gRPC along with the message format for request and response.
- Generating server and client code / stubs from the .proto file using grpcio-tools package in python
- Creating the branch.py file containing the following:
    - Query functionality
    - Deposit functionality
    - Withdraw functionality
    - propagate_interface functionality using gRPC.
    - For the propagate_transaction functionality it is important to call other branch stubs with the propagate flag set to False otherwise the program will be stuck in an infinite loop.
    - message_delivery functionality which reads the customer request and performs the action according to the interface/operation_type.
    - The message_delivery acquires a lock before updating the balance for a branch and releases the lock only when the update has been propagated to all the branches.
    - This prevents the customer from making read requests on their balance until their transaction has been propagated to all branches which ensures read-your-writes client-centric consistency.
- Creating the customer.py file containing the following:
    - execute_events functionality: This reads the events stored in the customers event list and calls the stub for the branch associated with the branch id in the request using gRPC. Once it gets the response from the branch, it then stores the response in the customer's recd_msgs.
    - get_customer_results: This method returns the results received by the customer in the format mentioned in the output in the project document.
- Creating the main.py file which does the following:
    - Create the empty output files.
    - Read and parse data from the input file.
    - Start the branch processes and initialize their stubs for communicating with other branches and other required data like id, balance and ids of other branches.
    - Start the cutomer processes and also initialize other required details like their id and events they have to process.
    - After a customer process has finished execution, gather its received messages by calling get_customer_results and add them to the output file.
    - Once all the customer processes are finished stop the branch servers so that the program exits gracefully.

Results

```
[
 {
   "id": 1,
   "recv": [
     {
       "interface": "query",
       "branch": 1,
       "balance": 0
     }
   ]
 },
 {
   "id": 1,
   "recv": [
     {
       "interface": "deposit",
       "branch": 1,
       "result": "success"
     }
   ]
 },
 {
   "id": 1,
   "recv": [
     {
       "interface": "query",
       "branch": 1,
       "balance": 10
     }
   ]
 },
 {
   "id": 1,
   "recv": [
     {
       "interface": "query",
       "branch": 2,
       "balance": 10
     }
   ]
 },
 {
   "id": 1,
   "recv": [
     {
       "interface": "deposit",
       "branch": 2,
      "result": "success"
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 2,
      "balance": 20
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 3,
      "balance": 20
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "deposit",
      "branch": 3,
      "result": "success"
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 3,
      "balance": 30
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 4,
        "balance": 30
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "deposit",
        "branch": 4,
        "result": "success"
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 4,
        "balance": 40
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 5,
        "balance": 40
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "deposit",
        "branch": 5,
        "result": "success"
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 5,
      "balance": 50
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 6,
      "balance": 50
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "deposit",
      "branch": 6,
      "result": "success"
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 6,
      "balance": 60
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 7,
      "balance": 60
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "deposit",
      "branch": 7,
      "result": "success"
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 7,
      "balance": 70
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 8,
      "balance": 70
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "deposit",
      "branch": 8,
      "result": "success"
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 8,
      "balance": 80
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 9,
        "balance": 80
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "deposit",
        "branch": 9,
        "result": "success"
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 9,
        "balance": 90
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 10,
        "balance": 90
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "deposit",
        "branch": 10,
        "result": "success"
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 10,
        "balance": 100
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 1,
        "balance": 100
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "withdraw",
        "branch": 1,
        "result": "success"
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 1,
        "balance": 90
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 2,
        "balance": 90
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "withdraw",
        "branch": 2,
        "result": "success"
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 2,
        "balance": 80
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 3,
        "balance": 80
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "withdraw",
        "branch": 3,
        "result": "success"
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 3,
        "balance": 70
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 4,
      "balance": 70
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "withdraw",
      "branch": 4,
      "result": "success"
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 4,
      "balance": 60
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 5,
      "balance": 60
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "withdraw",
      "branch": 5,
      "result": "success"
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 5,
        "balance": 50
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 6,
        "balance": 50
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "withdraw",
        "branch": 6,
        "result": "success"
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 6,
        "balance": 40
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 7,
        "balance": 40
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "withdraw",
        "branch": 7,
        "result": "success"
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 7,
        "balance": 30
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 8,
        "balance": 30
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "withdraw",
        "branch": 8,
        "result": "success"
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 8,
        "balance": 20
      }
    ]
  },
  {
    "id": 1,
    "recv": [
      {
        "interface": "query",
        "branch": 9,
      "balance": 20
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "withdraw",
      "branch": 9,
      "result": "success"
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 9,
      "balance": 10
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 10,
      "balance": 10
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "withdraw",
      "branch": 10,
      "result": "success"
    }
  ]
},
{
  "id": 1,
  "recv": [
    {
      "interface": "query",
      "branch": 10,
        "balance": 0
      }
    ]
  }
]
```

The implementation results are shown above. The justification for these results are as follows:

- Once a transaction has been made by a customer on a given branch we can see that the balance in next query operation on the branch where the update was made is the same as the balance of the query operation on any other branch.
- This shows that when the user updates their balance and moves to another branch to query their balance they get the updated result containing the last transaction that was last performed by them.
- This shows the banking system implementation offers a read-your-writes client-centric consistency model.
- We can see the balance gradually increase from 0 to 100 in steps of 10 after each deposit and it gradually decreases in steps of 10 from 100 to 0 after each withdraw operation. At all points along this process, a query is made to a branch where the balance is updated and to any other branch and both of the balances are same at each step.

This output facilitates the analysis of the distributed banking system's behavior and the correct enforcement of read-your-writes client-centric consistency model.