1. Design trade-off considered

For the concurrent Bank Server management, I had the following choices to consider

- Threaded server
- **Pre-Threaded server**
- Poll based server
- Epoll based server

However, I went ahead with the "Pre-Threaded" implementation because the pre-threaded server application beats the threaded server application in terms of performance and time per request required. Most of the web servers available in the market work on the same principle of "Pre-Threading". Thus, I have incorporated pre-threaded architecture on the server-side which is configurable.

Moreover, I have implemented the **Builder** (Customer.h, Transaction.h), **Observer** and **Singleton design pattern** (ObserverPattern.cpp) in the code to notify class that data has been received on the socket. So that transaction can be carried out. This helped me unnecessarily passing the class instances between the classes. This would have increased the coupling between the class. Using the above pattern helped me in decoupling the classes.

2. How synchronization is handled on the Server-side

In my implementation, I am using the HashMap to store the customer details (name, id, and balance).
I have implemented the **"Multiple Readers and Single Writer"** locking mechanism in my code.
`BankServer::get_customer_by_id` and `BankServer:: update_customer_by_id` methods implement reader/writer locking mechanism. This allows multiple readers to access HashMap concurrently whereas only one writer can do updates on the HashMap.
Also in thread-pool implementation threads are created before actually client starts connecting to the server. So how do you ensure thread synchronization(which thread is going to serve which client and accept call should be executed by single thread). In the `ServerSock::enter_server_loop` line no 82 mutex is added to ensure race conditions

3. Possible improvements in the application

- Maintain the transaction history of the customers and send them over a socket on-demand basis
- Run the Client application in the background and CLI in foreground
- Add support for vary request rate with respect to time i.e. send a request every 2s, 5s, etc.
- Verify why graph went down when the concurrency increased to more than 100. Mutex can be the reason. Using Linux perf tool or gdb we can debug the bottleneck.

4. Creativity added to the Project

- Added multithreading supported SPDILOG file and console logger. We are writing to file logs using this logger library. It is configured as a rolling file appender and console appender.
- Added design patterns in the project such as Observer, Singleton, and Builder
- Added Interest service in the Bank Server. Currently it is configured to 60 mins
- Used Apache AB tool to initially benchmark the Server performance and correctness
- Used C++ standard code design consideration. Usage **of separate header and implementation** file
- Adding reader/writer protection to standard **CPP unordered_map**

5. How to compile and run the server and client

- **Compiling and generating Server and Client executables**

  “make” or “make compile": this will generate both executables in the current directory

  “make clean”: this will delete all the object files and executables

- **Running the Server code :**

  The following are the command line options (CLI) available for Server. Please **make a sure** folder with name “logs” available in the same directory as executable
  Usage: ./Server [options]
  Options are:

  ```
  -t thread pool count   Thread pool count, default is 100
  -p port                Server port to listen, default is 8080
  -f file                Address of startup data file for customers, default is './src/Records.txt'
  ```

- **Running client code:**

  The following are the command line options (CLI) available for the Client. Please **make a sure** folder with name “logs” available in the same directory as executable
  Usage: ./Client [options]
  Options are:

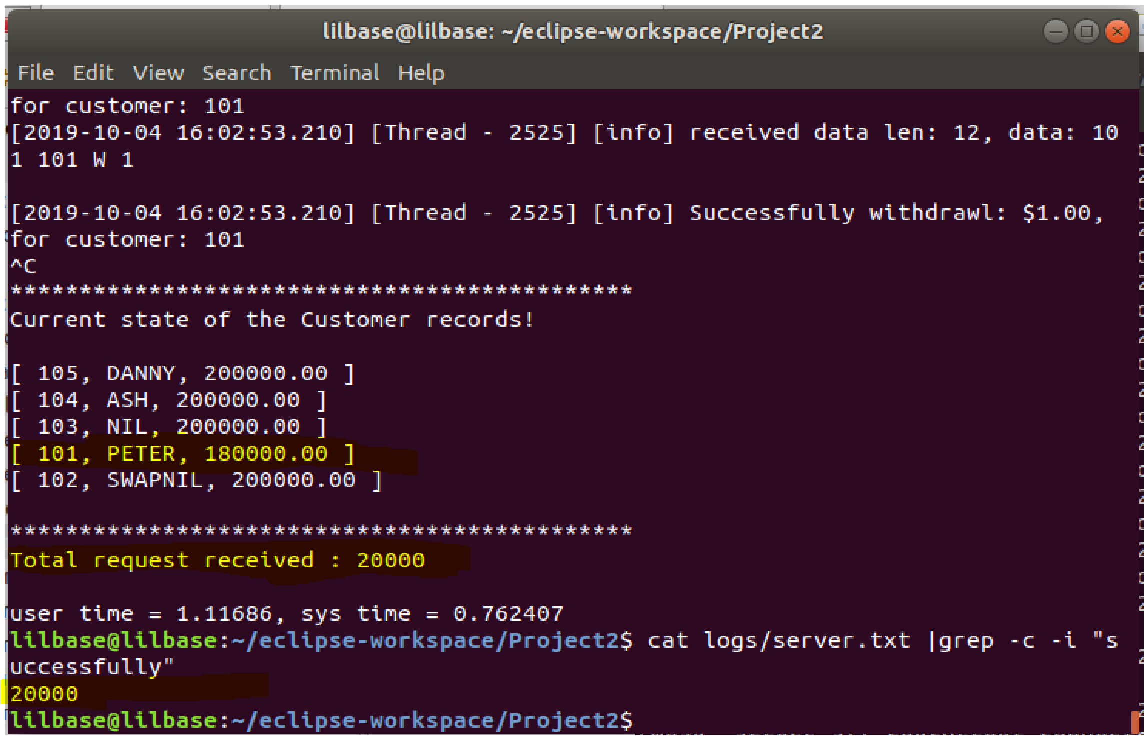
  ```
  -n requests      Number of requests to perform. Default is 20000
  -c concurrency   Number of multiple requests to make. Default is 30
  -h host          Host to connect to. Default is localhost
  -p port          Port to connect to. Default is 8080
  -f file          Address Transaction file address. Default is './src/Transactions.txt'
  ```

The output generated by the Both Server and client can be found in the logs directory.

Details/logs about the test run from the screenshot are in logs folder with the names as “server_1_2.txt “client_1_2.txt”, server_3_4.txt and client_3_4.txt”

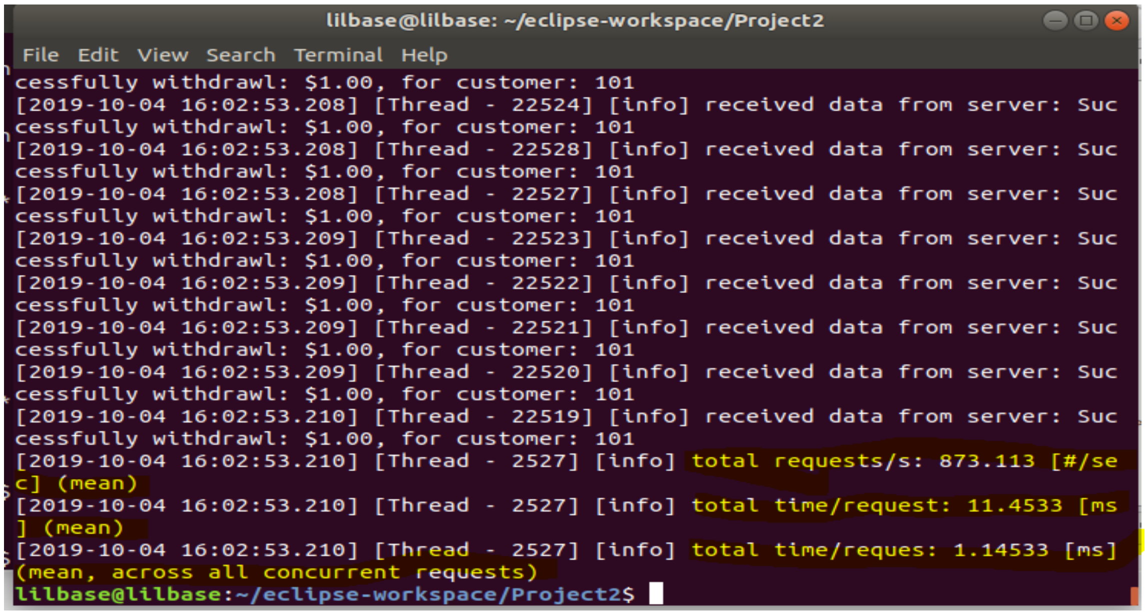
6. Some of the screenshot verifying correctness of the multithreaded server

1. Grep on client logs and server output for 20000 requests for withdrawal of $1 from account 101. Initial balance being $2,00,000, balance should remain as $1,80,000



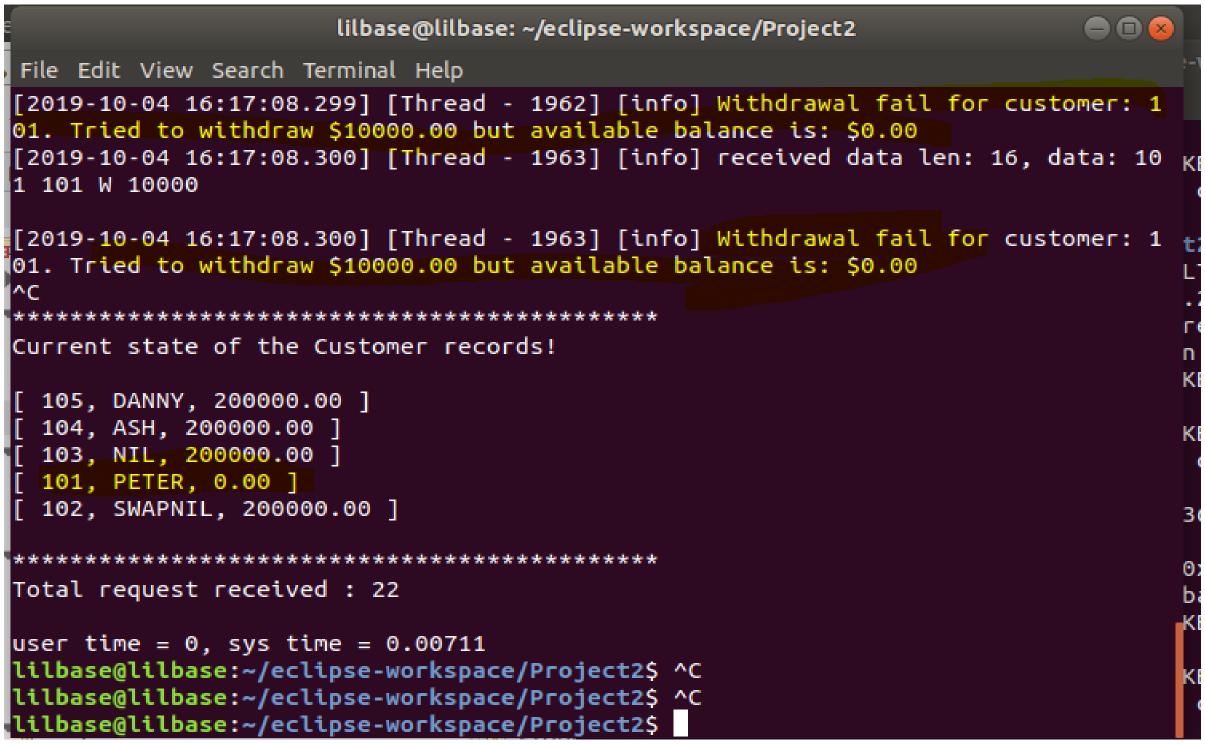
```
lilbase@lilbase: ~/eclipse-workspace/Project2
File Edit View Search Terminal Help
for customer: 101
[2019-10-04 16:02:53.210] [Thread - 2525] [info] received data len: 12, data: 10
1 101 W 1

[2019-10-04 16:02:53.210] [Thread - 2525] [info] Successfully withdrawl: $1.00,
for customer: 101
^C
***********************************************
Current state of the Customer records!

[ 105, DANNY, 200000.00 ]
[ 104, ASH, 200000.00 ]
[ 103, NIL, 200000.00 ]
[ 101, PETER, 180000.00 ]
[ 102, SWAPNIL, 200000.00 ]

***********************************************
Total request received : 20000

user time = 1.11686, sys time = 0.762407
lilbase@lilbase:~/eclipse-workspace/Project2$ cat logs/server.txt |grep -c -i "s
uccessfully"
20000
lilbase@lilbase:~/eclipse-workspace/Project2$
```

2. Client-side printing the benchmarking number after sending 20000 transaction msgs



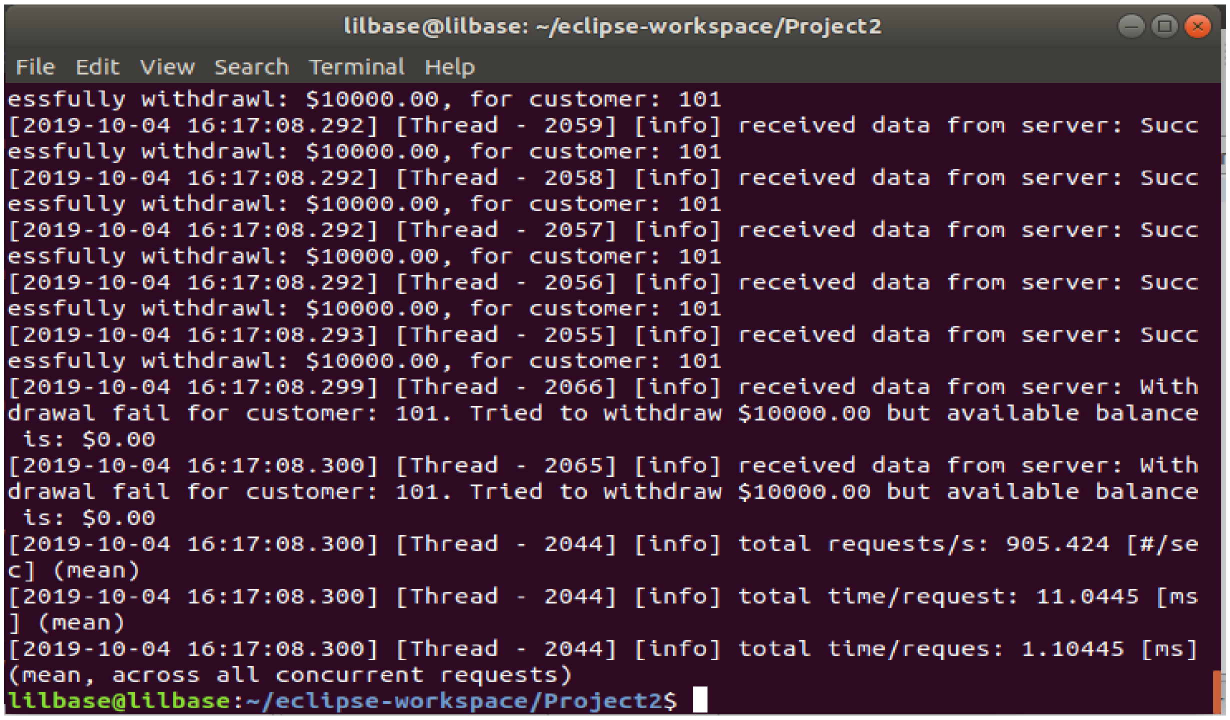
```
lilbase@lilbase: ~/eclipse-workspace/Project2
File Edit View Search Terminal Help
cessfully withdrawl: $1.00, for customer: 101
[2019-10-04 16:02:53.208] [Thread - 22524] [info] received data from server: Suc
cessfully withdrawl: $1.00, for customer: 101
[2019-10-04 16:02:53.208] [Thread - 22528] [info] received data from server: Suc
cessfully withdrawl: $1.00, for customer: 101
[2019-10-04 16:02:53.208] [Thread - 22527] [info] received data from server: Suc
cessfully withdrawl: $1.00, for customer: 101
[2019-10-04 16:02:53.209] [Thread - 22523] [info] received data from server: Suc
cessfully withdrawl: $1.00, for customer: 101
[2019-10-04 16:02:53.209] [Thread - 22522] [info] received data from server: Suc
cessfully withdrawl: $1.00, for customer: 101
[2019-10-04 16:02:53.209] [Thread - 22521] [info] received data from server: Suc
cessfully withdrawl: $1.00, for customer: 101
[2019-10-04 16:02:53.209] [Thread - 22520] [info] received data from server: Suc
cessfully withdrawl: $1.00, for customer: 101
[2019-10-04 16:02:53.210] [Thread - 22519] [info] received data from server: Suc
cessfully withdrawl: $1.00, for customer: 101
[2019-10-04 16:02:53.210] [Thread - 2527] [info] total requests/s: 873.113 [#/se
c] (mean)
[2019-10-04 16:02:53.210] [Thread - 2527] [info] total time/request: 11.4533 [ms
] (mean)
[2019-10-04 16:02:53.210] [Thread - 2527] [info] total time/reques: 1.14533 [ms]
(mean, across all concurrent requests)
lilbase@lilbase:~/eclipse-workspace/Project2$
```

3. Client initial balance is $2,00,000 and client is doing 22 transaction on server with each withdrawal is of $10000. Thus, server should return error for last two requests.



```
[2019-10-04 16:17:08.299] [Thread - 1962] [info] Withdrawal fail for customer: 1
01. Tried to withdraw $10000.00 but available balance is: $0.00
[2019-10-04 16:17:08.300] [Thread - 1963] [info] received data len: 16, data: 10
1 101 W 10000

[2019-10-04 16:17:08.300] [Thread - 1963] [info] Withdrawal fail for customer: 1
01. Tried to withdraw $10000.00 but available balance is: $0.00
^C
*********************************************
Current state of the Customer records!

[ 105, DANNY, 200000.00 ]
[ 104, ASH, 200000.00 ]
[ 103, NIL, 200000.00 ]
[ 101, PETER, 0.00 ]
[ 102, SWAPNIL, 200000.00 ]

*********************************************
Total request received : 22

user time = 0, sys time = 0.00711
lilbase@lilbase:~/eclipse-workspace/Project2$ ^C
lilbase@lilbase:~/eclipse-workspace/Project2$ ^C
lilbase@lilbase:~/eclipse-workspace/Project2$
```

4. On client side we also received error saying withdrawal failed



```
essfully withdrawl: $10000.00, for customer: 101
[2019-10-04 16:17:08.292] [Thread - 2059] [info] received data from server: Succ
essfully withdrawl: $10000.00, for customer: 101
[2019-10-04 16:17:08.292] [Thread - 2058] [info] received data from server: Succ
essfully withdrawl: $10000.00, for customer: 101
[2019-10-04 16:17:08.292] [Thread - 2057] [info] received data from server: Succ
essfully withdrawl: $10000.00, for customer: 101
[2019-10-04 16:17:08.292] [Thread - 2056] [info] received data from server: Succ
essfully withdrawl: $10000.00, for customer: 101
[2019-10-04 16:17:08.293] [Thread - 2055] [info] received data from server: Succ
essfully withdrawl: $10000.00, for customer: 101
[2019-10-04 16:17:08.299] [Thread - 2066] [info] received data from server: With
drawal fail for customer: 101. Tried to withdraw $10000.00 but available balance
 is: $0.00
[2019-10-04 16:17:08.300] [Thread - 2065] [info] received data from server: With
drawal fail for customer: 101. Tried to withdraw $10000.00 but available balance
 is: $0.00
[2019-10-04 16:17:08.300] [Thread - 2044] [info] total requests/s: 905.424 [#/se
c] (mean)
[2019-10-04 16:17:08.300] [Thread - 2044] [info] total time/request: 11.0445 [ms
] (mean)
[2019-10-04 16:17:08.300] [Thread - 2044] [info] total time/reques: 1.10445 [ms]
(mean, across all concurrent requests)
lilbase@lilbase:~/eclipse-workspace/Project2$
```