

Distributed Sequencer

- Siddhant Dilip Godshalwar

- **Sequencer**

The task is to create a distributed sequencer service that generates unique and sequentially increasing numerical ids for client requests. Each time a client sends a `get_id()` request, the sequencer service should provide a new id that is one greater than the previous id, ensuring that each id is unique and higher than the previous one.

- **Implementation**

I have implemented the code using 3 individual components:- sequencer, server, and client

Sequencer:-

- The Sequencer class is responsible for generating unique ids, which are sent to clients upon receiving a "get_id" request.
- The start method sets up a socket connection and generates an id to be broadcasted to available servers via the broadcast method.
- The broadcast method uses the `concurrent.futures.ThreadPoolExecutor` class to send the id to each server in a separate thread using the `tell_all` function.
- The `tell_all` function establishes a socket connection to each server and sends the id, which is sent back to the sequencer and then to the client.
- This ensures all servers have a consistent view of the latest id.

Server:-

- The Server class listens for connections on a specified host and port. Upon receiving a message with an id generated by the sequencer, the id, and the client's address are added to a priority queue.
- The start method sets up the connection, adds the received id to the queue, and calls the `request_handler` method.
- The `request_handler` sends back the next available id to the client if it matches the one at the top of the queue.

- The method checks if the queue is not empty and if the id at the top matches the next available id.
- The loop continues until the connection is closed.

Client:-

- This script connects to a server using a socket connection and prompts the user to enter 'get' to request a new unique id from the sequencer service, or 'quit' to exit the program.
- If the user enters 'get', the client sends a 'get' message to the server, receives the generated id, and prints it to the console.
- If the user enters 'quit', the program exits the while loop and terminates.

• Test Case

Servers: 48, Clients: 3

In the screenshot, the top left terminal is for the sequencer.py and the other 3 terminals represent an individual client. As seen, all ids being generated are in handled individually with no repetition.

This also works for a small number of servers showing same behavior and also for a large number of servers and clients.

```

OpenSSH SSH client
Sequencer is sending request for port 7021 from ('127.0.0.1', 50076)
Sequencer is sending request for port 7023 from ('127.0.0.1', 42582)
7036 Server is sending the id b'20'
Sequencer is sending request for port 7035 from ('127.0.0.1', 44548)
7038 Server is sending the id b'20'
Broadcasting to server 7044
7042 Server is sending the id b'20'
7041 Server is sending the id b'20'
7024 Server is sending the id b'20'
7045 Server is sending the id b'20'
Sequencer is sending request for port 7044 from ('127.0.0.1', 56768)
7026 Server is sending the id b'20'
7029 Server is sending the id b'20'
7035 Server is sending the id b'20'
7037 Server is sending the id b'20'
Broadcasting to server 7047
7032 Server is sending the id b'20'
7020 Server is sending the id b'20'
Sequencer is sending request for port 7047 from ('127.0.0.1', 52788)
7021 Server is sending the id b'20'
7023 Server is sending the id b'20'
7046 Server is sending the id b'20'
7039 Server is sending the id b'20'
7015 Server is sending the id b'20'
7047 Server is sending the id b'20'
7044 Server is sending the id b'20'

OpenSSH SSH client
Last login: Tue Apr  4 03:01:05 2023 from 68.50.221.6
sigods@silos:~$ cd repo/DistributedSystems/Sequencer/
sigods@silos:~/repo/DistributedSystems/Sequencer$ python3 client.py
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 1
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 3
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 4
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 8
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 16
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 18
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 19
Type 'get' to trigger get_id() or 'quit' to exit:

OpenSSH SSH client
Last login: Tue Apr  4 03:02:02 2023 from 68.50.221.6
sigods@silos:~$ cd repo/DistributedSystems/Sequencer/
sigods@silos:~/repo/DistributedSystems/Sequencer$ python3 client.py
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 2
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 7
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 9
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 14
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 15
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 20
Type 'get' to trigger get_id() or 'quit' to exit:

OpenSSH SSH client
Last login: Tue Apr  4 02:54:00 2023 from 68.50.221.6
sigods@silos:~$ cd repo/DistributedSystems/Sequencer/
sigods@silos:~/repo/DistributedSystems/Sequencer$ python3 client.py
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 5
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 6
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 10
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 11
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 12
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 13
Type 'get' to trigger get_id() or 'quit' to exit: get
Received id: 17
Type 'get' to trigger get_id() or 'quit' to exit:

```

- **How to Run**

Mentioned in the Readme file