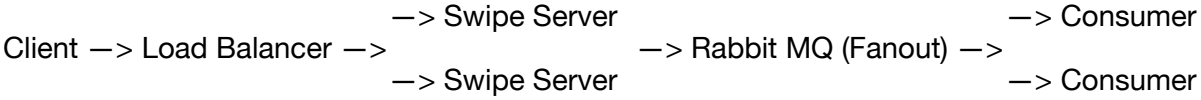
Repo: https://github.com/imsashahu/6650-a2

Design:



- Client (client.java) has main function to create a Timestamp object to record the start time of the execution, calls the executeInThreads method to create and start multiple threads, and then records the end time of the execution. It also calculates throughput. The executeInThreads method creates a CountDownLatch object with a count of numThreads and a List object to store the number of requests to be sent by each thread. It then creates a Runnable object for each thread and starts it on a new thread. After all threads have completed their tasks, it prints the total number of successful and unsuccessful requests and returns the number of successful requests.
- Client (MyRunnable.java) implements Runnable. It connects to server or load balancer of several servers, to send out all assigned number of requests in all threads successfully.
- Client (RandomGenerator.java) randomly generates requests that are valid.
- Client (Test.java) - Not In Use

——

- Swipe Server (swipe.java) The servlet first checks if the URL is valid based on its structure, and returns an appropriate response code and message if it is not valid. The servlet reads the request body and deserializes it into a Java object of type SwipeDetails using the Google Gson library. It then validates the data. If the data is valid, the servlet constructs a message payload in JSON format and publishes it to a RabbitMQ message queue using a shared connection and channel. Since we need to consume the queue in two different ways, I use fanout to send to 2 queues, with the same messages.
- Swipe Server (SwipeDetails.java) a class of swipe action. 3 private fields: swiper: an integer representing the ID of the user who swiped. swipee: an integer representing the ID of the user who was swiped. comment: a string representing any comment that the swiper left about the swipee.

——

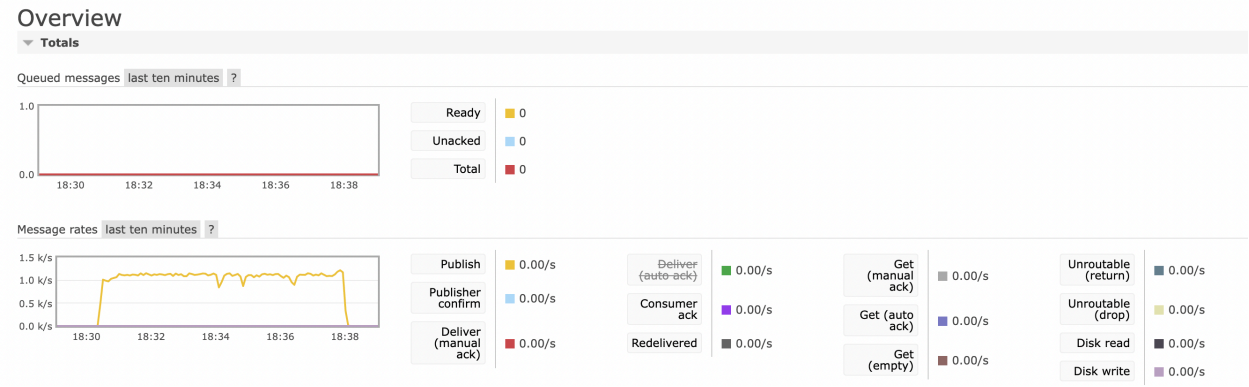
- Consumer (RabbitMQ.java) connects to a RabbitMQ broker using the host, username, and password specified in the code. It then creates 50 threads and runs a MyRunnable instance in each thread. The MyRunnable instances represent workers that will consume messages from the specified queue and store them in the gotten/given ConcurrentHashMap, “givingNumbers” is for the number of likes and dislikes this user has swiped, “givingList” is for a list of who the user has swiped right on. Overall, the code appears to be a simple implementation of a message queue system using RabbitMQ with multithreading support. Currently I choose a thread size of 50, which means for each thread, I open a channel for it to consume the message. Also, fanout is used in this script, for both consumers to consume from their queues separately.
- Consumer (MyRunnable.java) when a thread is created with an instance of MyRunnable, it establishes a connection to RabbitMQ, creates a channel to consume messages from the specified queue, and waits for messages to arrive. When a message arrives, it is deserialized

into an instance of the SwipeDetails class using Google's Gson library. The swipe object is then used to update the two hash maps that are multithread safe. Finally, the message is marked as delivered by invoking the basicConsume method on the channel object.

- Consumer (SwipeDetails.java) a class of swipe action. Same as in swipe server.

Report:

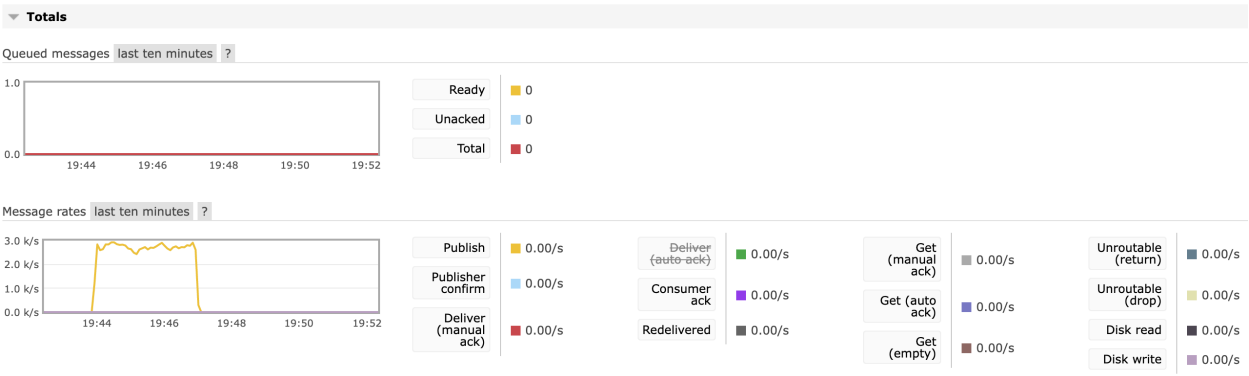
Without load balancer:




```
/Users/SashaHu/Library/Java/JavaVirtualMachines/corretto-11.0.18/Contents/Home/bin/java ...
Total number of successful requests is: 500000
Total number of unsuccessful requests is: 0
Total run time in milliseconds is: 458330
Total throughput in requests per second is: 1090

Process finished with exit code 0
```

With load balancer:




```
/Users/SashaHu/Library/Java/JavaVirtualMachines/corretto-11.0.18/Contents/Home/bin/java ...
Total number of successful requests is: 500000
Total number of unsuccessful requests is: 0
Total run time in milliseconds is: 185077
Total throughput in requests per second is: 2701

Process finished with exit code 0
```