## Github Repo

https://github.com/boweill/cs6650/tree/main/Assignment2

## Server Design

### Pkg: Bean (for plain Java Objects)

#### ResponseMsg

- Corresponds to the ResponseMsg schema specified in API spec (https://app.swaggerhub.com/apis/IGORTON/Twinder/1.0.0#);

#### SwipeDetails

- Corresponds to the SwipeDetails schema specified in API spec (https://app.swaggerhub.com/apis/IGORTON/Twinder/1.0.0#);
- Field 'action' added so that consumer can know whether the swiper liked or disliked swipe and use that information for data aggregation.

### Pkg: QueueUtils

#### RMQChannelFactory

- Responsible for constructing individual channels used to publish to the queues.
- Reference from text book Ch 7.

#### RMQChannelPool

- A fixed sized channel pool where the PublisherRunnable threads can borrow channels from to publish messages.
- In this server the channel pool has a size of 10;
- Reference from text book Ch 7.

### Pkg: Servlet

#### SwipeServlet

- The Servlet that handles requests from client.
- In its init() method it establishes connection to RabbitMQ and sets up a channel pool with a fixed amount of channels initialized;
- In its doPost() method it borrows a channel from its channel pool, publishes to a fanout exchange which will push the message to all the queues it's aware of, returns the channel to the pool, and send back a response in the format of "${swiper} liked / disliked ${swipee}".

## Consumer Design

### Pkg: Bean (for plain Java Objects)

#### DataAggregator

- The class that has two Maps in it to store the likes and dislikes of user.
- The likeData maps user to a Set of users they liked;

- The dislikeData maps user to the number of user the disliked (thought it unnecessary to know which users were disliked since our system does not include a recommendation functionality);
- Both maps are synchronized as the Consumer class has multiple threads updating it at the same time;
- Handles SwipeDetails by storing into likeData or dislikeData depending on user action.
- Can return the number of users the current user liked / disliked, also can return potential matches for current user by returning up to the first 100 users in the set of users that were liked.

#### SwipeDetails

- Corresponds to the SwipeDetails schema specified in API spec (https://app.swaggerhub.com/apis/IGORTON/Twinder/1.0.0#);
- Field 'action' added so that consumer can know whether the swiper liked or disliked swipe and use that information for data aggregation.

### Pkg: Consumer

#### LikeConsumer

- The class that establishes connection RabbitMQ and starts numOfThreads number of individual consumer threads that uses give exchangeName, queueName to consume from the queue.
- Contains a main method which starts a LikeConsumer object with the command line arguments (exchangeName, queueName, numOfThreads) it received.

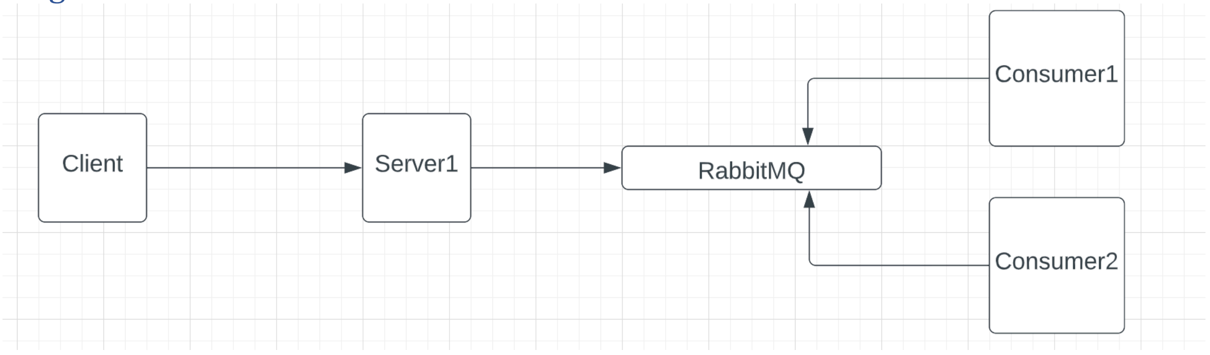
### Pkg: Handler

#### LikeHandlerRunnable

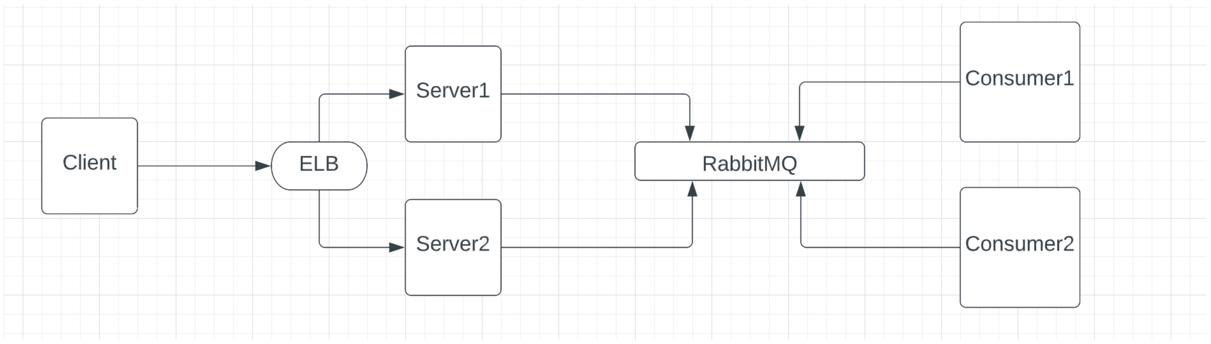
- The Runnable that provides the functionality for each individual consumer threads.
- In its run() method, it creates a channel from the connection initiated by LikeConsumer and registers a callback function to consume from the queue specified by the queueName passed in from LikeConsumer.

## Set Up

### Single Server



### Two servers behind a load balancer

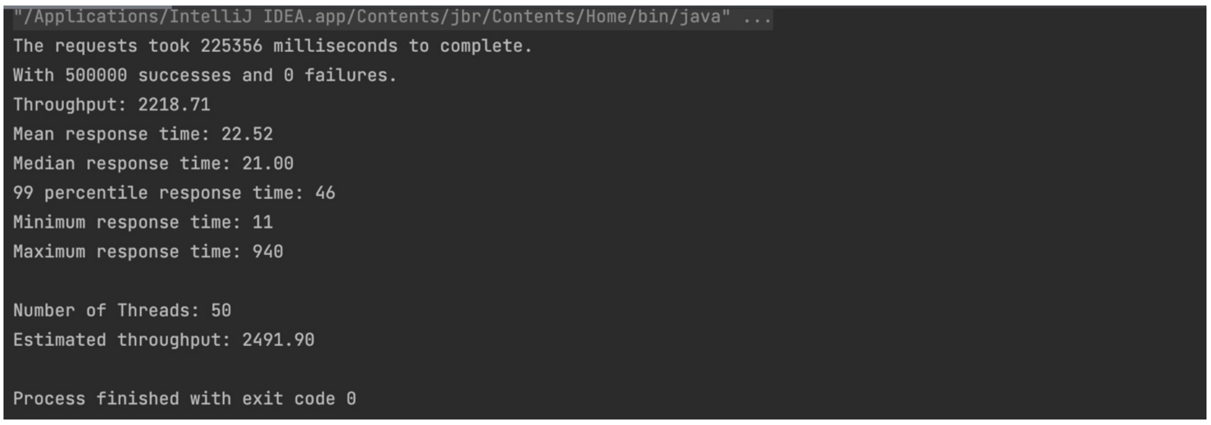


## Results

(Estimated throughput is acquired through sending 1000 requests with a single thread and calculating the estimate through Little's Law.)

### Single Server

#### 50 Client threads 200 Consumer threads

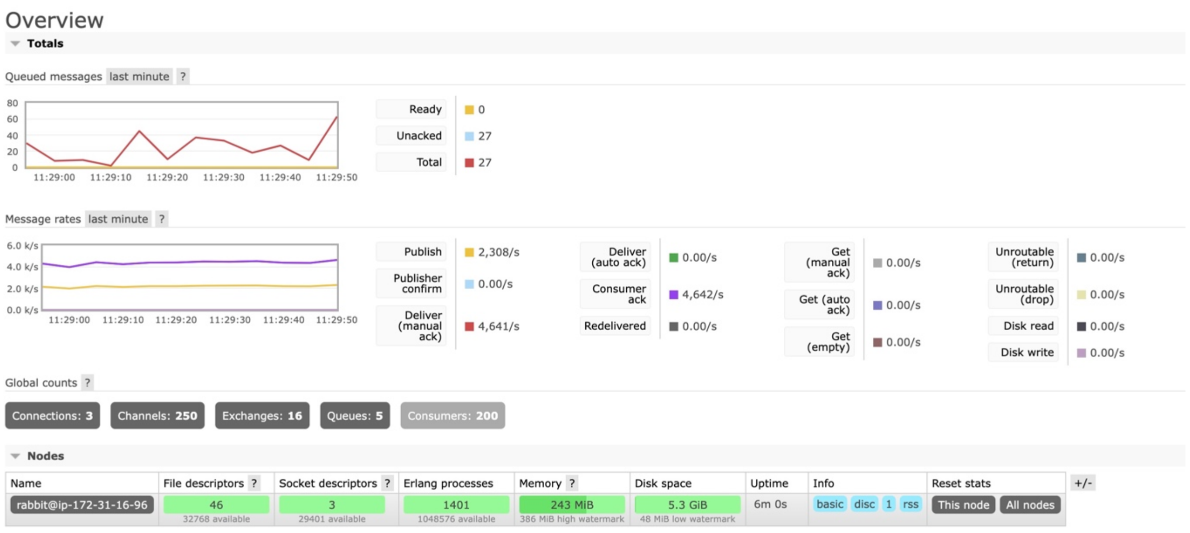


```
"/Applications/IntelliJ IDEA.app/Contents/jbr/Contents/Home/bin/java" ...
The requests took 225356 milliseconds to complete.
With 500000 successes and 0 failures.
Throughput: 2218.71
Mean response time: 22.52
Median response time: 21.00
99 percentile response time: 46
Minimum response time: 11
Maximum response time: 940

Number of Threads: 50
Estimated throughput: 2491.90

Process finished with exit code 0
```

Since we are using a fanout exchange with 2 queues bound we can see that the Deliver rate is roughly twice the publish rate.



Overview

Totals

Queued messages last minute ?

| | |
|---|---|
| Ready | 🟨 0 |
| Unacked | 🟦 27 |
| Total | 🟥 27 |

Message rates last minute ?

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Publish | 2,308/s | Deliver (auto ack) | 0.00/s | Get (manual ack) | 0.00/s | Unroutable (return) | 0.00/s |
| Publisher confirm | 0.00/s | Consumer ack | 4,642/s | Get (auto ack) | 0.00/s | Unroutable (drop) | 0.00/s |
| Deliver (manual ack) | 4,641/s | Redelivered | 0.00/s | Get (empty) | 0.00/s | Disk read | 0.00/s |
| | | | | | | Disk write | 0.00/s |

Global counts ?

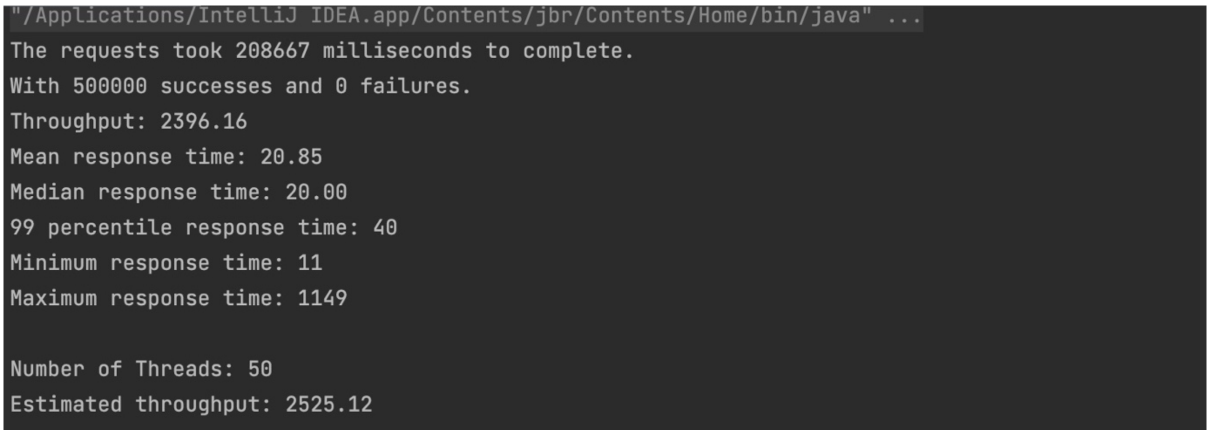
Connections: 3 Channels: 250 Exchanges: 16 Queues: 5 Consumers: 200

Nodes

| Name | File descriptors ? | Socket descriptors ? | Erlang processes | Memory ? | Disk space | Uptime | Info | Reset stats |
|---|---|---|---|---|---|---|---|---|
| rabbit@ip-172-31-16-96 | 46<br>32768 available | 3<br>29401 available | 1401<br>1048576 available | 243 MiB<br>386 MiB high watermark | 5.3 GiB<br>48 MiB low watermark | 6m 0s | basic disc 1 rss | This node All nodes |

## Two Servers behind Load Balancer

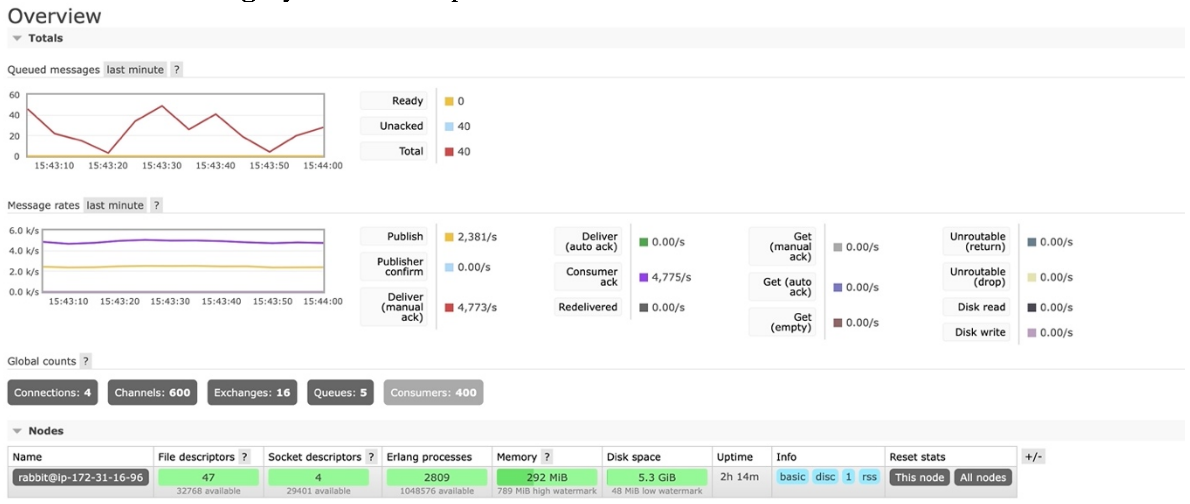
### 50 Client threads 200 Consumer threads



```
"/Applications/IntelliJ IDEA.app/Contents/jbr/Contents/Home/bin/java" ...
The requests took 208667 milliseconds to complete.
With 500000 successes and 0 failures.
Throughput: 2396.16
Mean response time: 20.85
Median response time: 20.00
99 percentile response time: 40
Minimum response time: 11
Maximum response time: 1149

Number of Threads: 50
Estimated throughput: 2525.12
```

Since we are using a fanout exchange with 2 queues bound we can see that the Deliver rate is roughly twice the publish rate.



Overview

Totals

Queued messages last minute ?

| | |
|---|---|
| Ready | 🟨 0 |
| Unacked | 🟦 40 |
| Total | 🟥 40 |

Message rates last minute ?

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Publish | 2,381/s | Deliver (auto ack) | 0.00/s | Get (manual ack) | 0.00/s | Unroutable (return) | 0.00/s |
| Publisher confirm | 0.00/s | Consumer ack | 4,775/s | Get (auto ack) | 0.00/s | Unroutable (drop) | 0.00/s |
| Deliver (manual ack) | 4,773/s | Redelivered | 0.00/s | Get (empty) | 0.00/s | Disk read | 0.00/s |
| | | | | | | Disk write | 0.00/s |

Global counts ?

Connections: 4 Channels: 600 Exchanges: 16 Queues: 5 Consumers: 400

Nodes

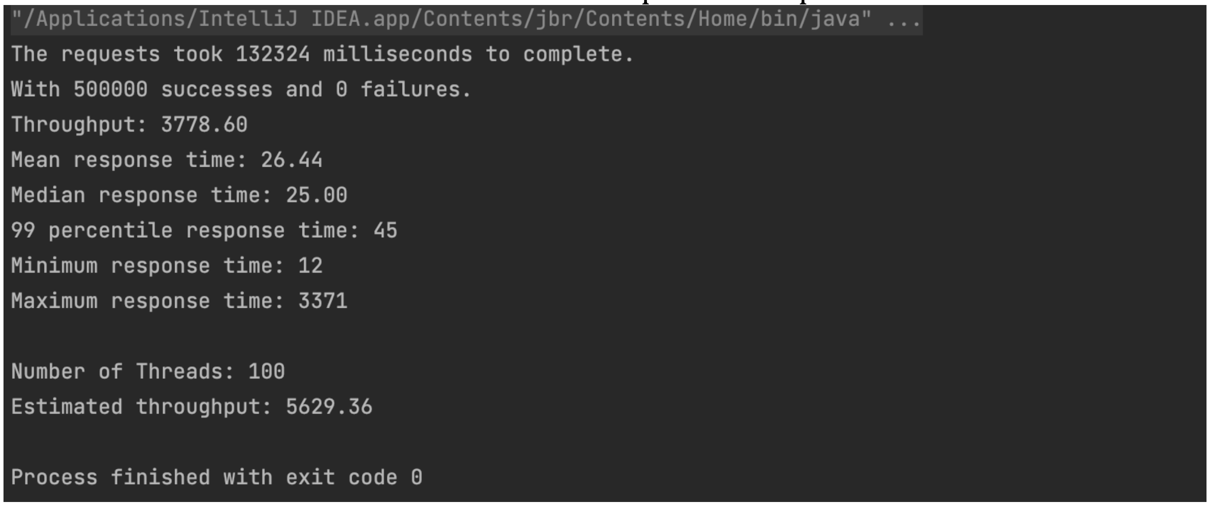
| Name | File descriptors ? | Socket descriptors ? | Erlang processes | Memory ? | Disk space | Uptime | Info | Reset stats |
|---|---|---|---|---|---|---|---|---|
| rabbit@ip-172-31-16-96 | 47<br>32768 available | 4<br>29401 available | 2809<br>1048576 available | 292 MiB<br>789 MiB high watermark | 5.3 GiB<br>48 MiB low watermark | 2h 14m | basic disc 1 rss | This node All nodes |

With the same client and consumer thread setup we see that load balancer results in a slightly higher throughput and steadier message rates.

## 100 Client threads 400 Consumer threads

To handle more client threads, I had to scale up the RabbitMQ server and run consumer jars on separate instances instead of the same one.

This without doubt resulted in higher throughput, but it is further from the throughput estimate calculated from single thread, which leads to the conclusion that around 50 client threads would be the optimal setup.

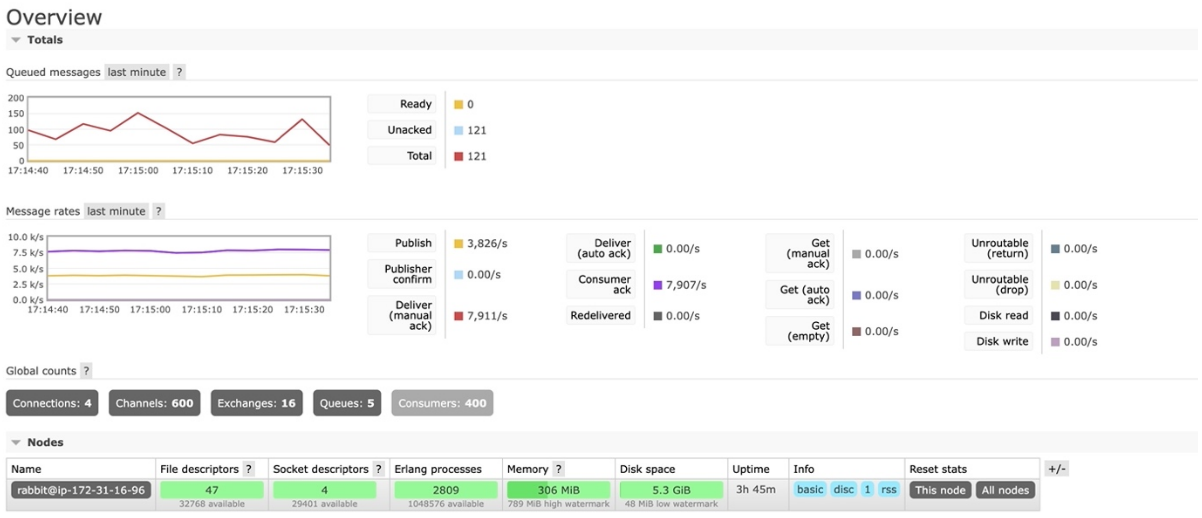


```
"/Applications/IntelliJ IDEA.app/Contents/jbr/Contents/Home/bin/java" ...
The requests took 132324 milliseconds to complete.
With 500000 successes and 0 failures.
Throughput: 3778.60
Mean response time: 26.44
Median response time: 25.00
99 percentile response time: 45
Minimum response time: 12
Maximum response time: 3371

Number of Threads: 100
Estimated throughput: 5629.36

Process finished with exit code 0
```

Since we are using a fanout exchange with 2 queues bound we can see that the Deliver rate is roughly twice the publish rate.



Overview

Totals

Queued messages last minute ?

| | |
|---|---|
| Ready | 🟨 0 |
| Unacked | 🟦 121 |
| Total | 🟥 121 |

Message rates last minute ?

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Publish | 🟨 3,826/s | Deliver (auto ack) | 🟩 0.00/s | Get (manual ack) | ⬜ 0.00/s | Unroutable (return) | ⬛ 0.00/s |
| Publisher confirm | 🟦 0.00/s | Consumer ack | 🟪 7,907/s | Get (auto ack) | 🟦 0.00/s | Unroutable (drop) | 🟨 0.00/s |
| Deliver (manual ack) | 🟥 7,911/s | Redelivered | ⬛ 0.00/s | Get (empty) | 🟫 0.00/s | Disk read | ⬛ 0.00/s |
| | | | | | | Disk write | 🟪 0.00/s |

Global counts ?

Connections: **4** Channels: **600** Exchanges: **16** Queues: **5** Consumers: **400**

Nodes

| Name | File descriptors ? | Socket descriptors ? | Erlang processes | Memory ? | Disk space | Uptime | Info | Reset stats |
|---|---|---|---|---|---|---|---|---|
| rabbit@ip-172-31-16-96 | 47<br>32768 available | 4<br>29401 available | 2809<br>1048576 available | 306 MiB<br>789 MiB high watermark | 5.3 GiB<br>48 MiB low watermark | 3h 45m | basic disc 1 rss | This node All nodes |

+/-