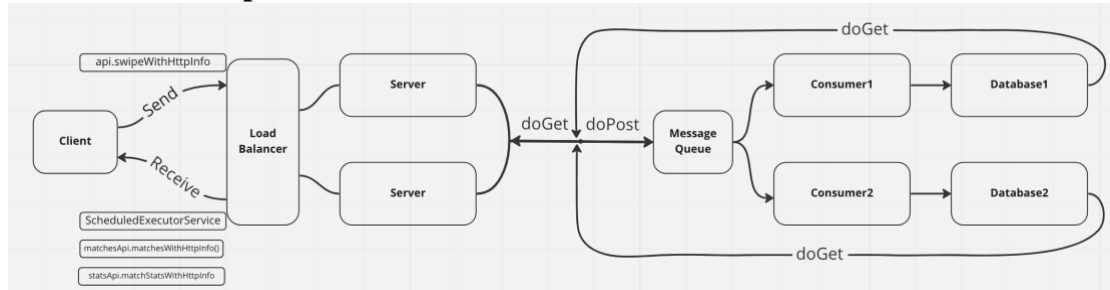


Assignment 3 Report Shengguo Zhou

URL: https://github.com/Shengguo-Zhou/CS6650_Distributed_System/tree/main

Flowchart Description:



For the assignment3, a data layer is needed. I put a Redis database behind the consumer to store the data passed from it. I implemented the doGet function inside the server. When the server will retrieve the data from database to the client. The client in assignment 3 can not only send data to server, but also can get data by calling the API from server. The details of each part is attached below:

The purpose of this report is to detail the design and implementation of a system for persisting swipe events and querying them. The solution consists of a client, a servlet, a message queue, a consumer, and a database for storing the swipe data.

Design: The solution architecture is designed to achieve high throughput and low response times while being simple and low cost. The client generates swipe events and sends them to a servlet, which writes them to a persistent queue called TempStore. A consumer reads from TempStore and updates a SwipeData database that stores information about users and swipe events. The servlet implements GET requests and retrieves results directly from the SwipeData database.

Servlet - The servlet implements doPost and writes every swipe event to TempStore. It also implements doGet and retrieves results from SwipeData.

Message Queue - TempStore is implemented as a message queue that ensures safe and persistent storage of the swipe events.

Consumer - The consumer reads from TempStore and updates Swipe Data. It ensures that the database is up to date with the latest swipe events.

Database - SwipeData is a database that stores information about users and swipe events. It is designed to enable highly efficient querying for the GET requests. In my assignment 3, I choose Redis as my database.

Client:

Main function is in “MultiThread” class. Inside this class:

First step, I created a queue and put 500k request inside it. And the queue is a global variable.

Nest step, I use a for loop to generate 200 threads to achieve multiply-thread. For each of thread, whenever this thread is started, it will pull a request from the queue. If the queue is empty, it will pull nothing, which means this thread would do nothing. So which thread is started first, which one may be tasked with more sending request tasks. Therefore, within this step, there would be 500k requests in total, and the 200 threads need to send it to the local or EC2 server. When all the requests are sent, which means the queue is empty, the 200 threads would then have a rest.

Last step, there would be some other global variable that would count the successful request number and the failed one, the lasting time and so on.

RunInOneThread.java: This class is designed for each one of the thread, I need to send the data through api.swipeWithHttpInfo. Inside the run function, the detail of each thread is implemted, queue is the global passed from the main function, when a new thread is created, it will get a request body from the queue. And it won't stop until the queue is empty. There are another two parameters to count the number of successful and failed sending tasks. In the sentEvent function, it will try at most 5 times to sent one request, if it is successful within 5 times of try, success will plus one. When it is failed, this request will be abandoned and failure will plus one.

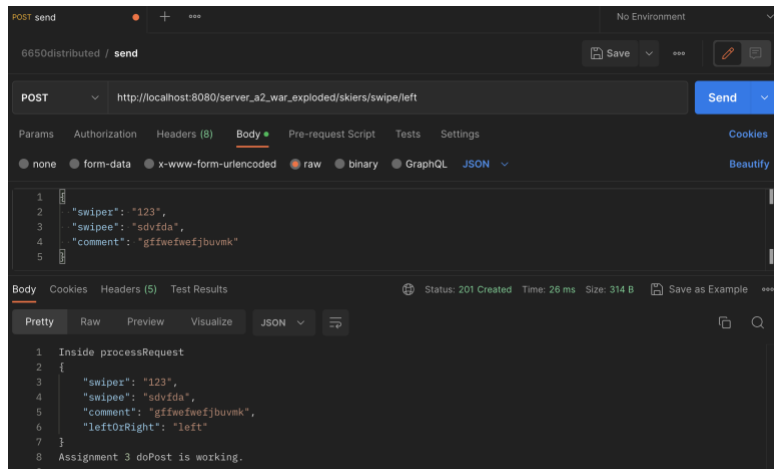
SwipeEvent.java inside the event file and BodyList.java and GenerateSingleBody.java is the helper function for main and singe-thread functions. For each of the request, I have packaged the info within the request and named it to a event class. When all the 500k request body is generated, it would be put into the list, to be specific, it is the queue that I have mentioned.

For the output part, I added nothing but result data processing class and a csv output file class. Inside the RunInOneThread.java class, when a new request is successfully sent, I will put the success info into another global list to store it. When all the requests are sent, I will get a list of 500k success info. And then I can sort this list by certain criteria. Like I can sort it by the delay time, then I can get the result of minimum, maximum, medium, 99%percentage response time. And like this, if I sort it by its start time, I can plot the chart by its beginning time to know for each of a certain second, how many requests would be sent.

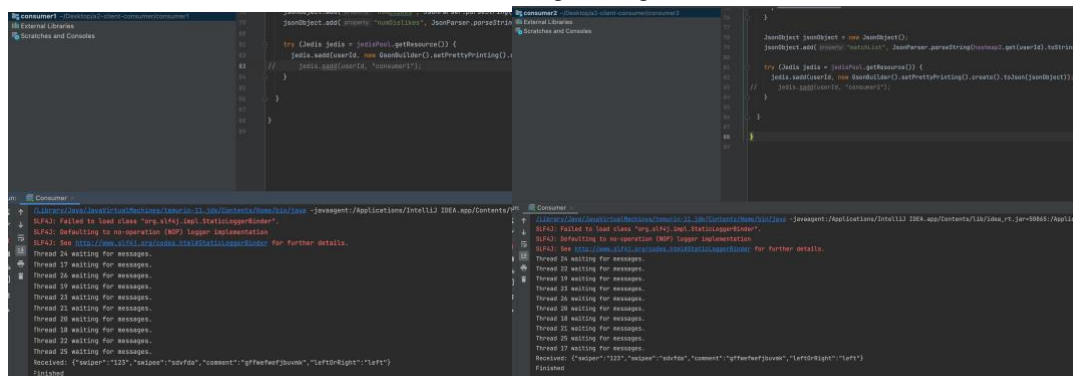
Result:

Linked the consumer with the redis database. When Postman send data to the server, the server would post it to the database, so if we get the keys inside the database, we could find the result.

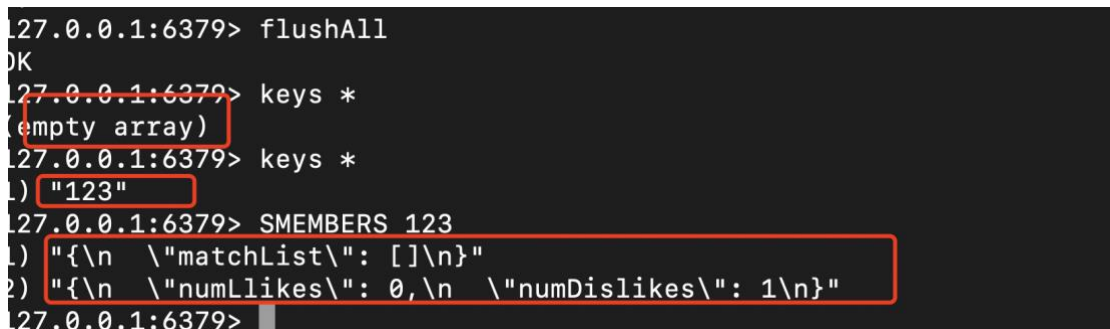
I started the testing from only 1 request which is from Postman
Postman:



Consumer 1 & 2 started with 10 threads for reciving messages:

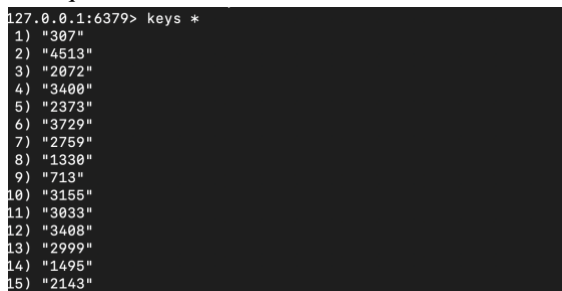


Redis:



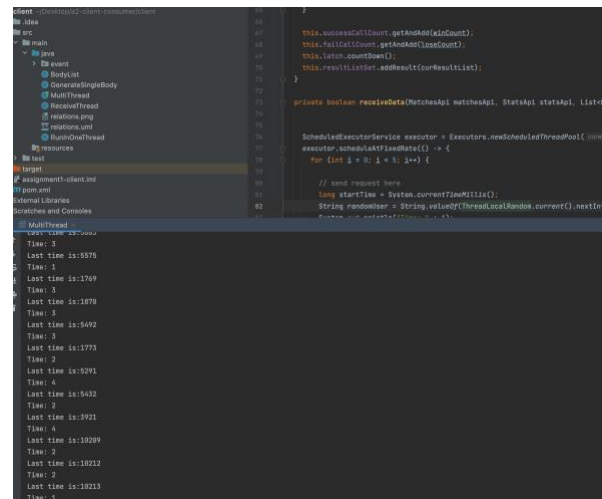
It works well, so I improved the requests number to 50.

50 requests in redis:



```
35) "566"  
36) "1449"  
37) "3289"  
38) "4643"  
39) "3255"  
40) "1771"  
41) "3826"  
42) "2348"  
43) "4382"  
44) "3798"  
45) "3878"  
46) "627"  
47) "2374"  
48) "338"  
49) "2413"  
50) "4642"  
  
127.0.0.1:6379> SMEMBERS 4642  
1) "(\\n \\matchList\\": [\\n 464147\\n ]\\n)"  
2) "(\\n \\numLikes\\": 1,\\n \\numDislikes\\": 0\\n)"  
127.0.0.1:6379> □
```

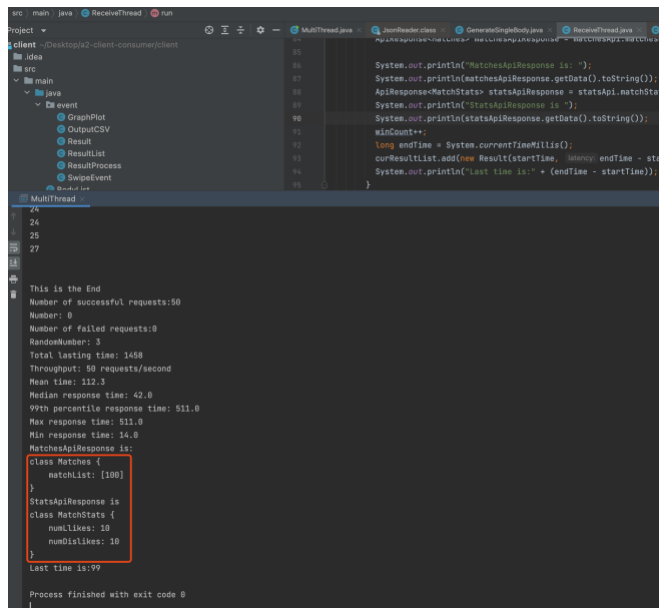
Client:



Message queue



Result returned from API



```
Project: client
src:
  main:
    java:
      Event
      OutputCSV
      Result
      ResultList
      ResultProcess
      SwapEvent
      MultiThread.java

MultiThread.java
74
24
25
27

This is the End
Number of successful requests:50
Number: 0
Number of failed requests:0
RandomNumber: 3
Total lasting time: 1458
Throughput: 50 requests/second
Mean time: 112.3
Median response time: 42.0
99th percentile response time: 511.0
Max response time: 511.0
Min response time: 14.0
MatchesApiResponse is:
{
  "matchList": [100]
}
StatsApiResponse is
{
  "numLikes": 10,
  "numDislikes": 10
}
Last time is:99
Process finished with exit code 0
```

It works well so then I keep on improving the number of requests:
Swiper and swipee are in range of [1, 100], 5000 requests generated in total.



It works well, then begin our test from Local side:

1.

500k requests,

swiper & swipee range from [1, 50000],

consumer 1 & 2 : 100 threads,

client: 50threads

Test 1

```
This is the End
Number of successful requests:500000
Number of failed requests:0
Total lasting time: 273914
Throughput: 1831 requests/second
Mean time: 12.81584
Median response time: 38.0
99th percentile response time: 62.0
Max response time: 968.0
Min response time: 0.0

This is the End in receiving data
Number of successful requests in receiving data:685
Number of failed requests in receiving data:0
Total lasting time in receiving data: 273914
Throughput in receiving data: 2 requests/second
Mean time in receiving data: 5.43216788321168
Median response time in receiving data: 2.0
99th percentile response time in receiving data: 39.0
Max response time in receiving data: 238.0
Min response time in receiving data: 1.0

Process finished with exit code 0
```

Test2

```
This is the End
Number of successful requests:500000
Number of failed requests:0
Total lasting time: 376509
Throughput: 1329 requests/second
Mean time: 8.94019
Median response time: 6.0
99th percentile response time: 49.0
Max response time: 1097.0
Min response time: 1.0

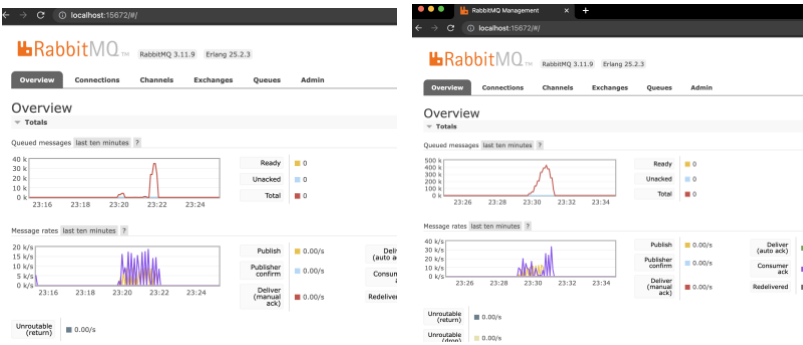
This is the End in receiving data
Number of successful requests in receiving data:1365
Number of failed requests in receiving data:0
Total lasting time in receiving data: 376509
Throughput in receiving data: 3 requests/second
Mean time in receiving data: 3.9501831501831504
Median response time in receiving data: 2.0
99th percentile response time in receiving data: 24.0
Max response time in receiving data: 238.0
Min response time in receiving data: 0.0

Process finished with exit code 0
```

Returned value from API and Response time:

```
RandomNumber: 40465
MatchesApiResponse is:
class Matches {
    matchList: [42110, 15304, 27072, 25948, 39111, 33408]
}
StatsApiResponse is
class MatchStats {
    numLikes: 16
    numDislikes: 6
}
Last time is:2
Number: 4
RandomNumber: 20498
MatchesApiResponse is:
class Matches {
    matchList: [32982, 26241, 45662, 40355, 47486, 27487, 47207, 11197, 30770, 5859]
}
StatsApiResponse is
class MatchStats {
    numLikes: 7
    numDislikes: 10
}
Last time is:2
```

Rabbitmq



Consumer

```
Consumer
Received: {"swipe": "10835", "swipee": "33805",
"comment
": "HdeCPpJhJlJsgcDmPYMANECWUUpGpSsCdFvaxSngVArzIfgBBXCHzQPPSdpPgSsSaZeSxogAuSeTnKnLzXclwEYfUdUkYD6xH0tERFLC2gdZhrduveqIginLuwQMortqoXrmqYezYEUjcdMAJyatLPq6JYjCwIVHnCgA0nJ0LqvhUsUChNtBFuxgIPg0vVvAY
NDpLcPqUWwKluTtCayIMevnTb", "LeftOrRight": "right"}
Received: {"swipe": "6671", "swipee": "11662",
"comment
": "DcseJqNaliqCKQPJcxYerArDvZuboaycU5DRLLVSUEAwVUAFLSkmludYXAouW1L5ySuAQ1URUSVgFqPtCherPuqztpIurXCdLU1DwdBozFBUhWdWaiHBD1BrrnAb1KoBxNozvQWv1LnJWwduyVCDBRgPgwEwBfffftZntQlCpgdfmcYLzWSNhrZTqBnpULBWh
LHKGAFuAEQMSCTNVLxvyZNahemHs1", "LeftOrRight": "right"}
Finished
Received: {"swipe": "46614", "swipee": "9714",
"comment
": "BwjPuNhyssqDkDc10onsnnhqYmeS0GRRLBrHyeEkyVjbdCkZDvsjhbbgrhZCazDXcDXQW1dXh1erhzHfSoNkPMQjeNnhZHzbJjDBs0zShecEoBgeuQwvDpXCtZWUyGebEearACDyHYdmULguJUTFaJ3D-sBQbL1SPU0gdqELXqWLuVfZhaGtNlHqLkZhhZKxX
LcaeB3LFxHqWfYbYrGtGUiQXcTchM", "LeftOrRight": "right"}
Received: {"swipe": "44149", "swipee": "31858",
"comment
": "1RbaxQKtziwBmggtTpyuAEHYbsjporfQpwaamQTYHioBAPbtgShfkaWegkvUYVLqSvKJTjInwVTHQDeQpQHAqRgaZeYpRaBEHuuUYQkyEHAkktVUJQZJumFkamyZRRFSLaAuVHsMvzPn3JtKpLmJLnEgP0zKwKRPvLKfyIZdUDVHvqPLRckmUYkLuYDkxUFpZn
ZmHBTmwknfQHySZgkvNaaMDXhDF", "LeftOrRight": "Left"}
Finished
Received: {"swipe": "24393", "swipee": "49629",
"comment
": "1Dn3YHYMYqhabHsDXUYeshYanubciUAFgYHsgUgpmPgZrTEBrxdYtwJzPFCWeRZifchSonuUWlaJpZlgqexFyWbqqaZAnarFREIYuZuqbVRPLTpeLpQcPdcdNrrIKoyBdvCnNrMQhymdLTPZLKJRFlyXZJPglbJjgmsSk0J3CfZHSyBsaJNXXbLrFFstDpLyBV
ZjddrTZB5XrXocntzjhGapBNFNggl", "LeftOrRight": "Left"}
Finished
```

Database:

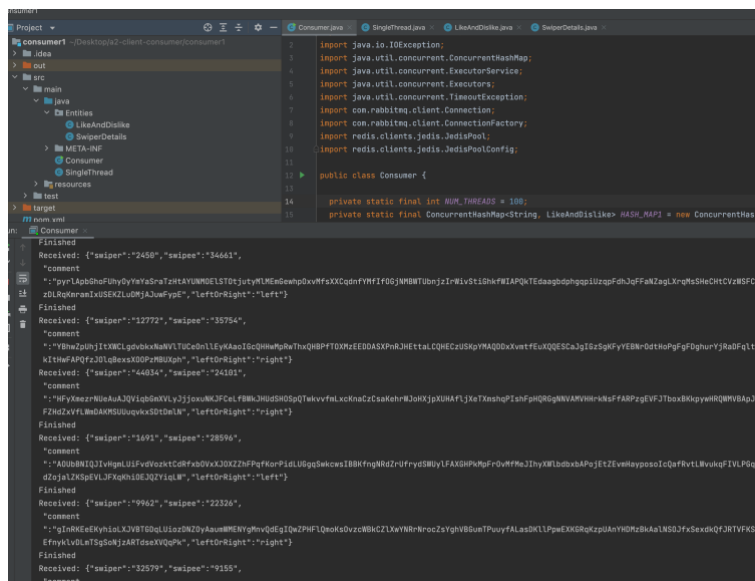
Table 0:

499761	"47879"
499771	"13281"
499781	"13379"
499791	"13385"
499801	"80895"
499811	"38829"
499821	"8288"
499831	"60805"
499841	"23687"
499851	"18898"
499861	"24555"
499871	"12765"
499881	"40950"
499891	"36419"
499901	"725"
499911	"27848"
499921	"22488"
499931	"48251"
499941	"18711"
499951	"26722"
499961	"32278"
499971	"34486"
499981	"5728"
127.0.0.1:6379[1]	

Table 1:

499781	"13372"
499791	"13335"
499801	"40895"
499811	"38829"
499821	"8288"
499831	"60805"
499841	"23687"
499851	"18898"
499861	"24555"
499871	"12765"
499881	"40950"
499891	"36419"
499901	"725"
499911	"27848"
499921	"22488"
499931	"48251"
499941	"18711"
499951	"26722"
499961	"32278"
499971	"34486"
499981	"5728"
127.0.0.1:6379[1]	get 5728
127.0.0.1:6379[1]	set 5728
127.0.0.1:6379[1]	

- 2.
- 500k requests,
- swipe & swipee range from[1, 50000],
- consumer 1 & 2 : 100 threads,
- client: 100 threads



```

49979) "40795"
49980) "45777"
49981) "29147"
49982) "12724"
49983) "7624"
49984) "41527"
49985) "16419"
49986) "44533"
49987) "26016"
49988) "22903"
49989) "30922"
49990) "26815"
49991) "49827"
49992) "13986"
49993) "33336"
49994) "34010"
49995) "44036"
49996) "38920"
49997) "38581"
49998) "47781"
49999) "20553"

127.0.0.1:6379> get 20553
"[4, 2]"

127.0.0.1:6379>

```

```
MultiThread
class MatchStats {
  numLikes: 4
  numDislikes: 12
}
Last time is:3

This is the End
Number of successful requests:500000
Number of failed requests:0
Total lasting time: 414124
Throughput: 1207 requests/second
Mean time: 25.984934
Median response time: 19.0
99th percentile response time: 119.0
Max response time: 4530.0
Min response time: 1.0

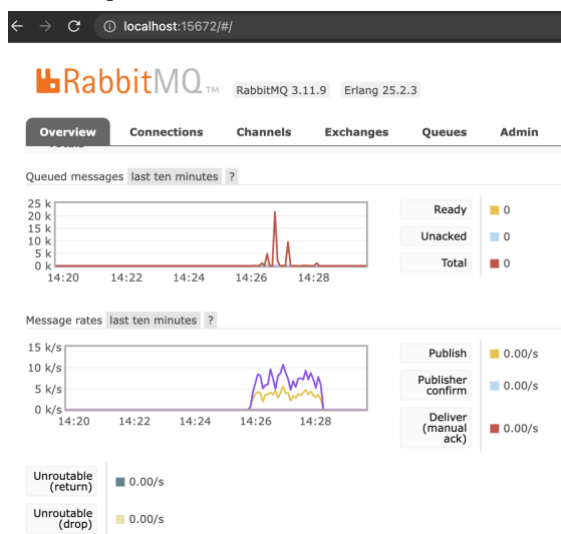
This is the End in receiving data
Number of successful requests in receiving data:1365
Number of failed requests in receiving data:0
Total lasting time in receiving data: 414124
Throughput in receiving data: 3 requests/second
Mean time in receiving data: 4.823443223443223
Median response time in receiving data: 3.0
99th percentile response time in receiving data: 31.0
Max response time in receiving data: 427.0
Min response time in receiving data: 0.0

Process finished with exit code 0
```

```
MultiThread
}
StatsApiResponse is
class MatchStats {
  numLikes: 3
  numDislikes: 4
}
Last time is:3
Number: 3
RandomNumber: 34893
MatchesApiResponse is:
class Matches {
  matchList: [3408, 44807]
}
StatsApiResponse is
class MatchStats {
  numLikes: 6
  numDislikes: 2
}
Last time is:3
Number: 4
RandomNumber: 31861
MatchesApiResponse is:
class Matches {
  matchList: [25429, 27769, 49413, 11615, 327, 20992, 8837, 8843, 13389, 18319, 38442, 456]
}
StatsApiResponse is
class MatchStats {
  numLikes: 4
  numDislikes: 12
}
Last time is:3
```

-
-
3.
500k requests,
swiper & swipe range from[1, 50000],
consumer 1 & 2 : 200 threads,
client: 200 threads

Rabbitmq




```

StatsApiResponse is
class MatchStats {
    numLikes: 4
    numDislikes: 6
}
Last time is:1
Number: 3
RandomNumber: 29667
MatchesApiResponse is:
class Matches {
    matchList: [27642, 27355, 14771, 23683, 30922, 29695]
}
StatsApiResponse is
class MatchStats {
    numLikes: 9
    numDislikes: 6
}
Last time is:1
Number: 4
RandomNumber: 43503
MatchesApiResponse is:
class Matches {
    matchList: [5002, 48220, 41032]
}
StatsApiResponse is
class MatchStats {
    numLikes: 6
    numDislikes: 3
}
Last time is:2

```

The screenshot shows an IDE with the following components:

- Project View:** A tree structure on the left showing a project named 'client' with subdirectories 'src' and 'main'. The 'src' directory contains a 'java' package with various classes like 'GraphPlot', 'OutputCSV', 'Result', 'ResultList', 'ResultProcess', 'SwingEvent', 'BodyList', 'GenerateSingleBody', 'MultiThread', and 'ReceiveThread'.
- Code Editor:** The 'MultiThread.java' file is open, showing a public class with static variables for request counts, a main method, and comments for IP and URL.
- Run Console:** The output of the program is displayed, showing statistics for the execution. Two sections are highlighted with red boxes:
 - Top Section:**

```

This is the End
Number of successful requests:500000
Number of failed requests:0
Total lasting time: 284374
Throughput: 1760 requests/second
Mean time: 48.45742
Median response time: 39.0
99th percentile response time: 198.0
Max response time: 3397.0
Min response time: 0.0

```
 - Bottom Section:**

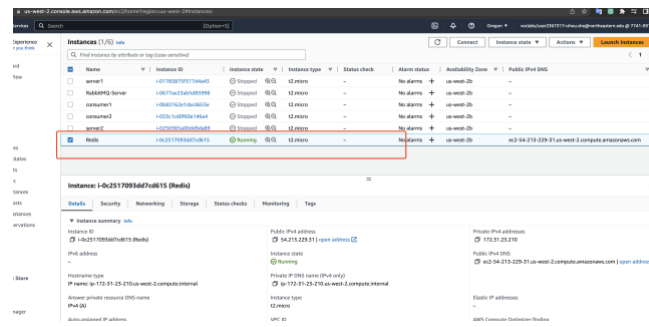
```

This is the End in receiving data
Number of successful requests in receiving data:768
Number of failed requests in receiving data:0
Total lasting time in receiving data: 284374
Throughput in receiving data: 2 requests/second
Mean time in receiving data: 5.818421052631579
Median response time in receiving data: 3.0
99th percentile response time in receiving data: 35.0
Max response time in receiving data: 537.0
Min response time in receiving data: 0.0

```
- Status Bar:** At the bottom, it says 'Process finished with exit code 0'.

The following part are tests from AWS side:

A new instance for Redis and give the access to it.



```
Setting up redis-server (6:7.0.10-1rl1-jammy) ...
Setting up redis (6:7.0.10-1rl1-jammy) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-29-148:~$ brew services start redis
Command 'brew' not found, did you mean:
  Command 'brew' from deb brew (0.4.1-1build1)
  Command 'brew' from deb bplay (0.991-10build1)
Try: sudo apt install <deb name>
ubuntu@ip-172-31-29-148:~$ redis-cli
127.0.0.1:6379> keys *
(empty array)
127.0.0.1:6379>
```

Ip address:

Database redis: 18.236.136.63

Consumer1: 52.11.54.7

Consumer2: 54.213.114.185

Rabbitmq: 35.166.166.121

Server1: 35.166.157.115

Server2: 54.212.9.5

loadBalancer: a3-lb-709645041.us-west-2.elb.amazonaws.com

Database redis: 34.213.111.105

Consumer1: 54.202.50.200

Consumer2: 54.186.31.145

Rabbitmq: 18.237.80.111

Server1: 35.86.147.140

Server2: 34.222.61.141

loadBalancer: a3-lb-709645041.us-west-2.elb.amazonaws.com

Database redis: 34.219.132.175

Consumer1: 54.200.33.85

Consumer2: 54.202.70.33

Rabbitmq: 54.201.155.64

Server1: 35.92.44.101

Server2: 52.10.99.67

loadBalancer: a3-lb-709645041.us-west-2.elb.amazonaws.com

```

[ec2-user@ip-172-31-31-213 ~]$ ls
consumer2.jar
[ec2-user@ip-172-31-31-213 ~]$ java -jar consumer2.jar
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Thread 67 waiting for messages.
Thread 117 waiting for messages.
Thread 163 waiting for messages.
Thread 116 waiting for messages.
Thread 115 waiting for messages.
Thread 114 waiting for messages.
Thread 113 waiting for messages.
Thread 112 waiting for messages.
Thread 111 waiting for messages.

```

```

6650distributed_system -- ec2-user@ip-172-31-25-235:~ -- ssh -i shengguo_key.cer ec2-user@52.11.54.7 -- 80x24
-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or directory
[ec2-user@ip-172-31-25-235 ~]$ ls
consumer1.jar
[ec2-user@ip-172-31-25-235 ~]$ java -jar consumer1.jar
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Thread 22 waiting for messages.
Thread 140 waiting for messages.
Thread 139 waiting for messages.
Thread 138 waiting for messages.
Thread 137 waiting for messages.
Thread 193 waiting for messages.
Thread 127 waiting for messages.
Thread 125 waiting for messages.

```

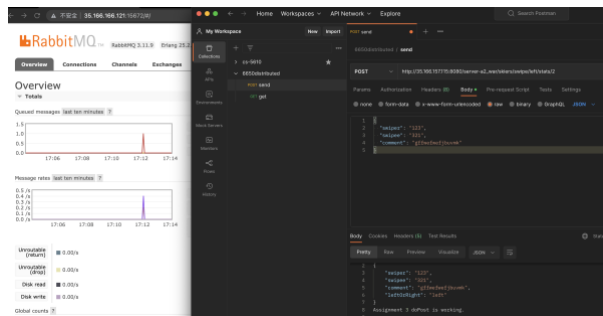
Pass the .war file to server (name is unchanged)

```

ec2-user@ip-172-31-21-39 tomcat]$ ls
BUILDING.txt    LICENSE    README.md    RUNNING.txt  conf    logs    webapps
CONTRIBUTING.md NOTICE    RELEASE-NOTES bin          lib     temp    work
ec2-user@ip-172-31-21-39 tomcat]$ cd webapps/
ec2-user@ip-172-31-21-39 webapps]$ ls
00T docs examples host-manager manager server-a2_war.war
ec2-user@ip-172-31-21-39 webapps]$ ls
00T docs examples host-manager manager server-a2_war server-a2_war.war
ec2-user@ip-172-31-21-39 webapps]$

```

1 request from Postman



```

Thread 35 waiting for messages.
Thread 36 waiting for messages.
Thread 33 waiting for messages.
Thread 32 waiting for messages.
Thread 38 waiting for messages.
Thread 29 waiting for messages.
Thread 28 waiting for messages.
Thread 27 waiting for messages.
Thread 26 waiting for messages.
Thread 26 waiting for messages.
Thread 23 waiting for messages.
Thread 22 waiting for messages.
Thread 21 waiting for messages.
Thread 20 waiting for messages.
Thread 19 waiting for messages.
Thread 18 waiting for messages.
Thread 17 waiting for messages.
Thread 16 waiting for messages.
Received: ("swipe": "123", "swipee": "321", "comment": "gffewefjbuuvk", "leftOrRight": "left")
Finished

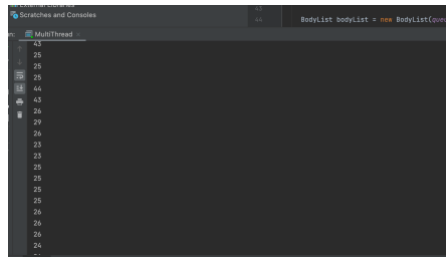
Thread 208 waiting for messages.
Thread 207 waiting for messages.
Thread 203 waiting for messages.
Thread 202 waiting for messages.
Thread 201 waiting for messages.
Thread 200 waiting for messages.
Thread 199 waiting for messages.
Thread 198 waiting for messages.
Thread 197 waiting for messages.
Thread 196 waiting for messages.
Thread 195 waiting for messages.
Thread 194 waiting for messages.
Thread 193 waiting for messages.
Thread 192 waiting for messages.
Thread 191 waiting for messages.
Thread 190 waiting for messages.
Thread 189 waiting for messages.
Thread 188 waiting for messages.
Thread 187 waiting for messages.
Thread 186 waiting for messages.
Thread 185 waiting for messages.
Thread 184 waiting for messages.
Thread 183 waiting for messages.
Thread 182 waiting for messages.
Thread 181 waiting for messages.
Thread 180 waiting for messages.
Thread 179 waiting for messages.
Thread 178 waiting for messages.
Thread 177 waiting for messages.
Thread 176 waiting for messages.
Thread 175 waiting for messages.
Thread 174 waiting for messages.
Thread 173 waiting for messages.
Thread 172 waiting for messages.
Thread 171 waiting for messages.
Thread 170 waiting for messages.
Thread 169 waiting for messages.
Thread 168 waiting for messages.
Thread 167 waiting for messages.
Thread 166 waiting for messages.
Thread 165 waiting for messages.
Thread 164 waiting for messages.
Thread 163 waiting for messages.
Thread 162 waiting for messages.
Thread 161 waiting for messages.
Thread 160 waiting for messages.
Thread 159 waiting for messages.
Thread 158 waiting for messages.
Thread 157 waiting for messages.
Thread 156 waiting for messages.
Thread 155 waiting for messages.
Thread 154 waiting for messages.
Thread 153 waiting for messages.
Thread 152 waiting for messages.
Thread 151 waiting for messages.
Thread 150 waiting for messages.
Thread 149 waiting for messages.
Thread 148 waiting for messages.
Thread 147 waiting for messages.
Thread 146 waiting for messages.
Thread 145 waiting for messages.
Thread 144 waiting for messages.
Thread 143 waiting for messages.
Thread 142 waiting for messages.
Thread 141 waiting for messages.
Thread 140 waiting for messages.
Thread 139 waiting for messages.
Thread 138 waiting for messages.
Thread 137 waiting for messages.
Thread 136 waiting for messages.
Thread 135 waiting for messages.
Thread 134 waiting for messages.
Thread 133 waiting for messages.
Thread 132 waiting for messages.
Thread 131 waiting for messages.
Thread 130 waiting for messages.
Thread 129 waiting for messages.
Thread 128 waiting for messages.
Thread 127 waiting for messages.
Thread 126 waiting for messages.
Thread 125 waiting for messages.
Thread 124 waiting for messages.
Thread 123 waiting for messages.
Thread 122 waiting for messages.
Thread 121 waiting for messages.
Thread 120 waiting for messages.
Thread 119 waiting for messages.
Thread 118 waiting for messages.
Thread 117 waiting for messages.
Thread 116 waiting for messages.
Thread 115 waiting for messages.
Thread 114 waiting for messages.
Thread 113 waiting for messages.
Thread 112 waiting for messages.
Thread 111 waiting for messages.
Thread 110 waiting for messages.
Thread 109 waiting for messages.
Thread 108 waiting for messages.
Thread 107 waiting for messages.
Thread 106 waiting for messages.
Thread 105 waiting for messages.
Thread 104 waiting for messages.
Thread 103 waiting for messages.
Thread 102 waiting for messages.
Thread 101 waiting for messages.
Thread 100 waiting for messages.
Thread 99 waiting for messages.
Thread 98 waiting for messages.
Thread 97 waiting for messages.
Thread 96 waiting for messages.
Thread 95 waiting for messages.
Thread 94 waiting for messages.
Thread 93 waiting for messages.
Thread 92 waiting for messages.
Thread 91 waiting for messages.
Thread 90 waiting for messages.
Thread 89 waiting for messages.
Thread 88 waiting for messages.
Thread 87 waiting for messages.
Thread 86 waiting for messages.
Thread 85 waiting for messages.
Thread 84 waiting for messages.
Thread 83 waiting for messages.
Thread 82 waiting for messages.
Thread 81 waiting for messages.
Thread 80 waiting for messages.
Thread 79 waiting for messages.
Thread 78 waiting for messages.
Thread 77 waiting for messages.
Thread 76 waiting for messages.
Thread 75 waiting for messages.
Thread 74 waiting for messages.
Thread 73 waiting for messages.
Thread 72 waiting for messages.
Thread 71 waiting for messages.
Thread 70 waiting for messages.
Thread 69 waiting for messages.
Thread 68 waiting for messages.
Thread 67 waiting for messages.
Thread 66 waiting for messages.
Thread 65 waiting for messages.
Thread 64 waiting for messages.
Thread 63 waiting for messages.
Thread 62 waiting for messages.
Thread 61 waiting for messages.
Thread 60 waiting for messages.
Thread 59 waiting for messages.
Thread 58 waiting for messages.
Thread 57 waiting for messages.
Thread 56 waiting for messages.
Thread 55 waiting for messages.
Thread 54 waiting for messages.
Thread 53 waiting for messages.
Thread 52 waiting for messages.
Thread 51 waiting for messages.
Thread 50 waiting for messages.
Thread 49 waiting for messages.
Thread 48 waiting for messages.
Thread 47 waiting for messages.
Thread 46 waiting for messages.
Thread 45 waiting for messages.
Thread 44 waiting for messages.
Thread 43 waiting for messages.
Thread 42 waiting for messages.
Thread 41 waiting for messages.
Thread 40 waiting for messages.
Thread 39 waiting for messages.
Thread 38 waiting for messages.
Thread 37 waiting for messages.
Thread 36 waiting for messages.
Thread 35 waiting for messages.
Thread 34 waiting for messages.
Thread 33 waiting for messages.
Thread 32 waiting for messages.
Thread 31 waiting for messages.
Thread 30 waiting for messages.
Thread 29 waiting for messages.
Thread 28 waiting for messages.
Thread 27 waiting for messages.
Thread 26 waiting for messages.
Thread 25 waiting for messages.
Thread 24 waiting for messages.
Thread 23 waiting for messages.
Thread 22 waiting for messages.
Thread 21 waiting for messages.
Thread 20 waiting for messages.
Thread 19 waiting for messages.
Thread 18 waiting for messages.
Thread 17 waiting for messages.
Thread 16 waiting for messages.
Thread 15 waiting for messages.
Thread 14 waiting for messages.
Thread 13 waiting for messages.
Thread 12 waiting for messages.
Thread 11 waiting for messages.
Thread 10 waiting for messages.
Thread 9 waiting for messages.
Thread 8 waiting for messages.
Thread 7 waiting for messages.
Thread 6 waiting for messages.
Thread 5 waiting for messages.
Thread 4 waiting for messages.
Thread 3 waiting for messages.
Thread 2 waiting for messages.
Thread 1 waiting for messages.

```

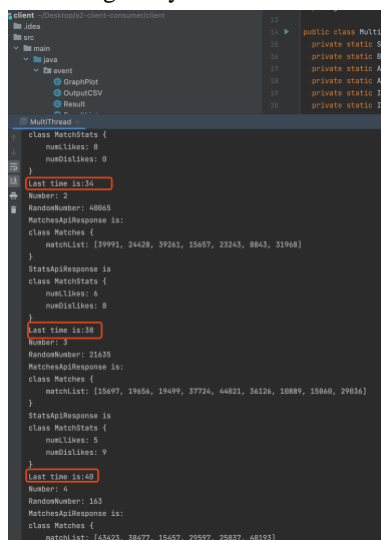
1.
 500k requests,
 swiper & swipee range from [1, 50000],
 consumer 1 & 2 : 200 threads,
 client: 50 threads

Clinet:

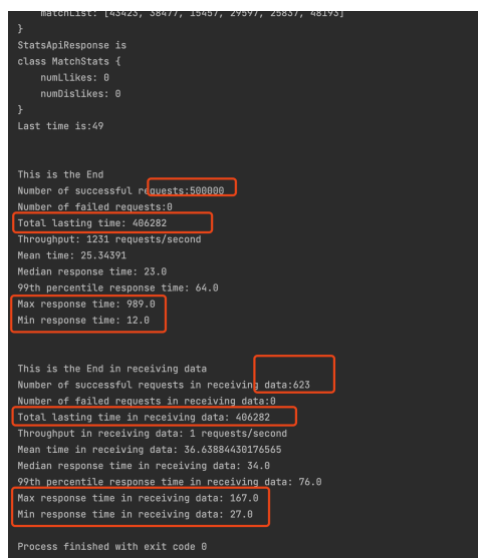
Sending data's delay time is around 20ms – 40ms



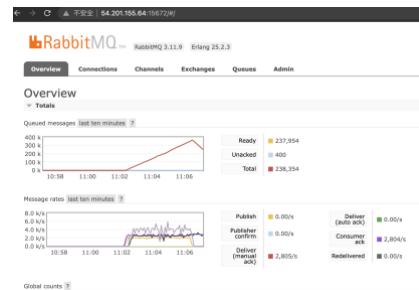
Receiving delay time and the result got after calling the two required API



Final result:



Turning point is shown:

[illegible]

499040	"34857"
499401	"30356"
499421	"26167"
499431	"27619"
499444	"13841"
499455	"35656"
499466	"41827"
499477	"20858"
499488	"10380"
499499	"10891"
499500	"17637"
499511	"28414"
499521	"43653"
499531	"25468"
499544	"49416"
499555	"22180"
499566	"35991"
499577	"37160"
499588	"34065"
499599	"14128"
499600	"48119"
499611	"31856"
499621	"17070"
499631	"20240"


```
ubuntu@ip-172-31-29-148:~$ redis-cli
127.0.0.1:6379> keys *
(empty array)
127.0.0.1:6379> select 1
OK
```

```
6650distributed_system -- ubuntu@ip-172-31-29-148:~$
49913) "44091"
49914) "7002"
49915) "42325"
49916) "48786"
49917) "32586"
49918) "26065"
49919) "8752"
49920) "24138"
49921) "20130"
49922) "20799"
49923) "7708"
49924) "21274"
49925) "34391"
49926) "35255"
49927) "34138"
49928) "47468"
49929) "39423"
49930) "22536"
49931) "49927"
49932) "8987"
49933) "7039"
49934) "3327"
(0.57s)
127.0.0.1:6379[1]>
```

2.

500k requests,

swiper & swipee range from[1, 50000],

consumer 1 & 2 : 200 threads,

client: 100 threads

client:

```
BodyList 19 pp
GenerateSingleRndv 20 pp
MultiThread
  numLikes: 9
  numDislikes: 17
  Last time is:36
  Number: 2
  RandomNumber: 4354
  MatchesApiResponse is:
  class Matches {
    matchList: [45172, 26415, 6531, 49528, 170, 6955, 31544]
  }
  StatsApiResponse is
  class MatchStats {
    numLikes: 12
    numDislikes: 9
  }
  Last time is:32
  Number: 3
  RandomNumber: 23421
  MatchesApiResponse is:
  class Matches {
    matchList: [27939, 2480, 27780, 36614, 16128, 25230, 30970, 2024]
  }
  StatsApiResponse is
  class MatchStats {
    numLikes: 11
    numDislikes: 11
  }
  Last time is:33
  Number: 4
  RandomNumber: 270
  MatchesApiResponse is:
```

```

class MatchStats {
    numLikes: 14
    numDislikes: 9
}
Last time is:36
Answer: 1
RandomNumber: 1019
MatchesApiResponse is:
class Matches {
    matchList: [25563, 29627, 36851, 6575, 16731, 8280, 46647, 28867, 24352, 34194, 27451, 27069, 1813, 46968, 2420, 14]
}
StatsApiResponse is:
class MatchStats {
    numLikes: 9
    numDislikes: 17
}
Last time is:36

```

```

BodyList
GenerateSingleBody
MultiThread
numLikes: 10
numDislikes: 10
}
Last time is:32

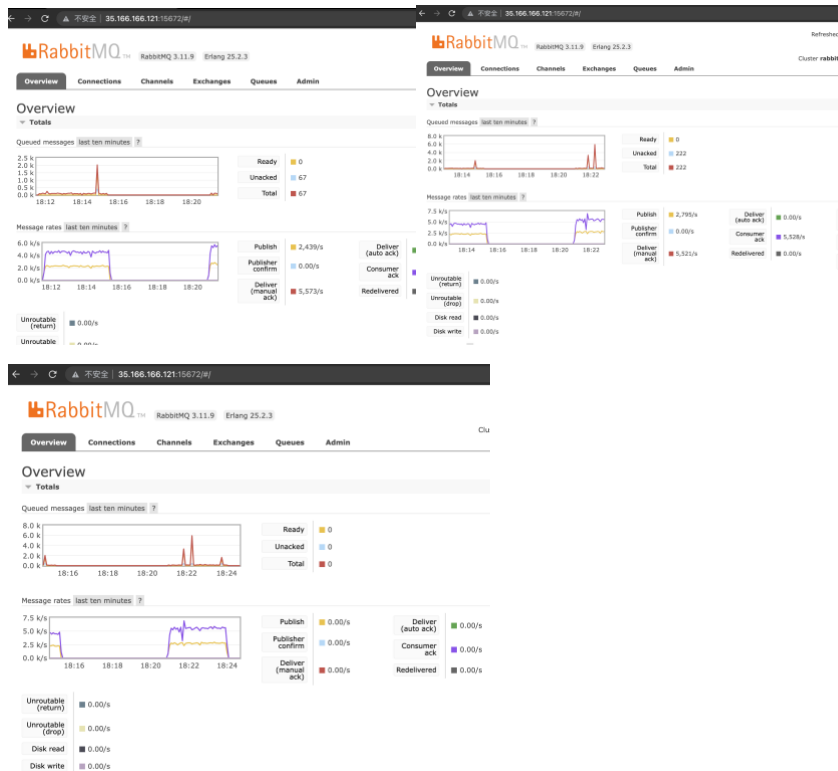
This is the End
Number of successful requests:500000
Number of failed requests:0
Total lasting time: 332274
Throughput: 1506 requests/second
Mean time: 43.921226
Median response time: 40.0
99th percentile response time: 112.0
Max response time: 1700.0
Min response time: 14.0

This is the End in receiving data
Number of successful requests in receiving data:450
Number of failed requests in receiving data:0
Total lasting time in receiving data: 332274
Throughput in receiving data: 1 requests/second
Mean time in receiving data: 39.34
Median response time in receiving data: 37.0
99th percentile response time in receiving data: 78.0
Max response time in receiving data: 157.0
Min response time in receiving data: 27.0

Process finished with exit code 0

```

Rabbitmq



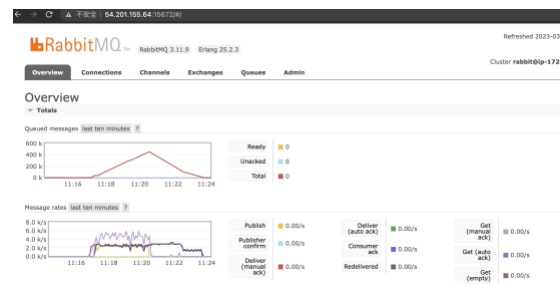
RabbitMQ 3.11.9 Overview

Queued messages (last ten minutes)

Ready	Unacked	Total
1,225,501	400	1,221,001

Message rates (last ten minutes)

Publish	Publisher confirm	Deliver (not ack)	Deliver (ack)
0.00/s	0.00/s	2.86/s	2.86/s



consumer

[illegible]

Redis

```
6650distributed_system — ubuntu
49950) "3020"
49951) "9540"
49952) "4387"
49953) "44193"
49954) "49553"
49955) "21666"
49956) "49210"
49957) "43558"
49958) "46553"
49959) "36700"
49960) "7236"
49961) "38571"
49962) "25606"
49963) "41742"
49964) "5967"
49965) "25600"
49966) "23094"
49967) "32209"
49968) "8220"
49969) "44631"
49970) "42149"
49971) "38979"
49972) "45466"
49973) "29974"
```

3.

500k requests,

swiper & swipee range from [1, 50000],

consumer 1 & 2 : 200 threads,

client: 200 threads

client:

```
MultiThread
numOfLikes: 18
Last time is: 34
Number: 1
RandomNumber: 29757
MatchesApiResponse is:
class Matches {
  matchList: [37555, 46276, 3762, 14899, 10446, 22086, 34875, 12596, 572, 38429, 23278, 30105]
}
StatsApiResponse is
class MatchStats {
  numLikes: 18
  numDislikes: 14
}
Last time is: 36
Number: 2
RandomNumber: 22647
MatchesApiResponse is:
class Matches {
  matchList: [47975, 41198, 1229, 27917, 5947, 21536, 20340, 20020, 9860, 9328, 25283, 40997, 11291]
}
StatsApiResponse is
class MatchStats {
  numLikes: 14
  numDislikes: 13
}
Last time is: 37
Number: 3
RandomNumber: 14378
MatchesApiResponse is:
class Matches {
  matchList: [39127, 36216, 49530, 25662, 949, 4721, 14908, 23742, 17657, 29607, 30534, 21435]
}
StatsApiResponse is
class MatchStats {
  numLikes: 14
  numDislikes: 13
}

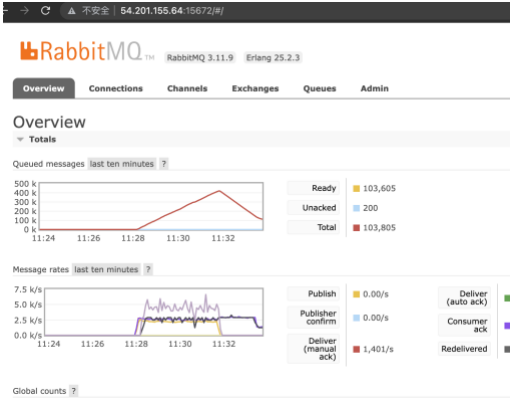
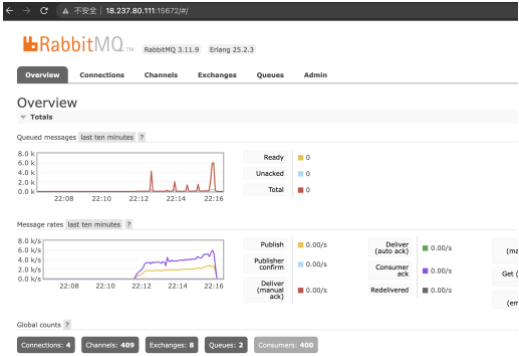
This is the End
Number of successful requests: 500000
Number of failed requests: 0
Total lasting time: 343868

Throughput: 1457 requests/second
Mean time: 91.159602
Median response time: 87.0
99th percentile response time: 167.0
Max response time: 2458.0
Min response time: 14.0

This is the End in receiving data
Number of successful requests in receiving data: 465
Number of failed requests in receiving data: 0
Total lasting time in receiving data: 343868
Throughput in receiving data: 1 requests/second
Mean time in receiving data: 38.5505376344086
Median response time in receiving data: 36.0
99th percentile response time in receiving data: 74.0
Max response time in receiving data: 122.0
Min response time in receiving data: 29.0

Process finished with exit code 0
```

Rabbitmq



consumer

```
6650distributed_system -- ec2-user@ip-172-31-31-213:~ -- ssh -i shengguo_key.cer ec2-user@54.213.114.185 -- 114x22
wflPjFzFqIfidiSSTHfSOrSZceyxWnRJyeEhJaIrhuuOfLzjyGAZsHzQUpwHGbuRmDFrcncWaoEGKQAmKEDk", "leftOrRight": "left"}
Finished
Received: {"swiper": "40141", "swipee": "41236", "comment": "IvKfYBBZZIRItCdvaZLKNHNUKiosafmFyvsnbEglorLxkvKjDQrQDQzMK
cwXQlLgpMqzMSitmsfuhRDmVWCIMGYroFDDkCqrvMzaDzGkSCZYeOOnxUrnppDggyXecrlHwjFStPgNnQKoolDxSAPJyhzMUGtdSJuesxklxSfGQS
qiZBHXaHuueuyjirdBjJwmkNPkjzmylJzoVXCBVJUGggqTybWKChTucBqnEMAggulwvIETZlsIgUtQKIwZH", "leftOrRight": "left"}
Received: {"swiper": "32524", "swipee": "33754", "comment": "mRpiZGEnOqaLyKvHmkmjmDKuaokELaggQMHBXoiwTsDyaUizJOaFquYzsz
GkPqjZgOIKlKJoQIascnNjQLABPvTxklgYhaDuveHYFSAcSLSEBihUEXFwGGcYMcXrlyPaTEpQwQPDUKvRsQOBmyQaOQkPTgfeHqzvYGtyvWulWC
dwjnSCpnkHAxtztjCdssXEdBOWzwsGjcECuInPLZthwcaQEIwHuDOrSyZsuCpdxTajjzZjpUIMbKqJmKTORe", "leftOrRight": "right"}
Finished
Received: {"swiper": "2822", "swipee": "39641", "comment": "xBThTHuUCPurCphMjdabEpVRDTlqsbOpALRjfeMzoBmOVrvIJcbLVxhROsv
nrtxpNIXpQfKgJtufbqkivSNpUYGKBZtNRlpTmJIWcbSCmxVdioBKPOMaSLJGSYodKQeJbIeWVPfxVCiAPfZgTzdXaHsyLUlQAxNVlHGDIdJWKTyqD
nGtqpAVuccDQVHsMivDJaXRSjnMidYCVFeqatTxMCDWYUJCJUBfUHXjePvMxIcfuxtCzOnkKgnEWEYkRYRa", "leftOrRight": "left"}
Finished
Received: {"swiper": "4157", "swipee": "8557", "comment": "waOJLHXgMqAHVCPDYzZngmVXUgSctUisnXHHUxwZDdPhQgIASaVTTcXgIBPC
AKAqzqbFPQwbeYCrzdQHydDjMlcsKWoEPjlxEXSIYbvLkwcCwqkQFVNjAzPEgEoBbTfJSTonASYJxPewBjAkIAOSyBKHWveOnyKMOYxdnwIhMJUCQ
AKodLYaLeIdMblvPWhjGInGiOPjSfTKbPNaKbLAcARBRfLrCsehUMXAOJODUpukxGffNymvuyfyLXVar", "leftOrRight": "right"}
Finished
```

Redis

```
49976) "41979"  
49977) "37599"  
49978) "34598"  
49979) "16461"  
49980) "19176"  
49981) "22450"  
49982) "30952"  
49983) "12604"  
49984) "43143"  
49985) "42269"  
49986) "43807"  
49987) "20057"  
49988) "5991"  
49989) "12612"  
49990) "25110"  
49991) "35276"  
49992) "17516"  
49993) "49799"  
49994) "39004"  
49995) "3984"  
49996) "16987"  
49997) "30260"  
49998) "8457"  
127.0.0.1:6379>
```

```
ubuntu@ip-172-31-29-148:~$ redis-cli ping  
PONG  
ubuntu@ip-172-31-29-148:~$ redis-cli  
127.0.0.1:6379> keys *  
(empty array)  
127.0.0.1:6379> keys *  
(empty array)
```

```
6650distributed_system — ubuntu  
49981) "10757"  
49982) "48786"  
49983) "42325"  
49984) "32586"  
49985) "7002"  
49986) "26065"  
49987) "8752"  
49988) "24138"  
49989) "20130"  
49990) "7708"  
49991) "20799"  
49992) "21274"  
49993) "34391"  
49994) "35255"  
49995) "47468"  
49996) "34138"  
49997) "39423"  
49998) "22536"  
49999) "49927"  
50000) "8987"  
50001) "7039"  
50002) "3327"  
(0.52s)  
127.0.0.1:6379>
```

Each part also works well on AWS EC2 Instance

Results: My solution was able to achieve throughput and latencies at the client within *10% of Assignment 2. The GET request mean latency was also not longer than the mean POST request latency. (36ms < 78ms in medium response time among the best performance results). *with the comparision with Report 2: [link](#)

Conclusion: In conclusion, my solution provides a reliable and efficient way of persisting swipe events and querying them. By utilizing a message queue and a consumer, I was able to ensure safe and persistent storage of the swipe events while keeping latencies low. My choice of database also enabled highly efficient querying for the GET requests. Overall, I was able to achieve results that met the requirements of the assignment while keeping the architecture and deployment simple and low cost.