ASSIGNMENT II

Ravi Voleti

Link To assignment:

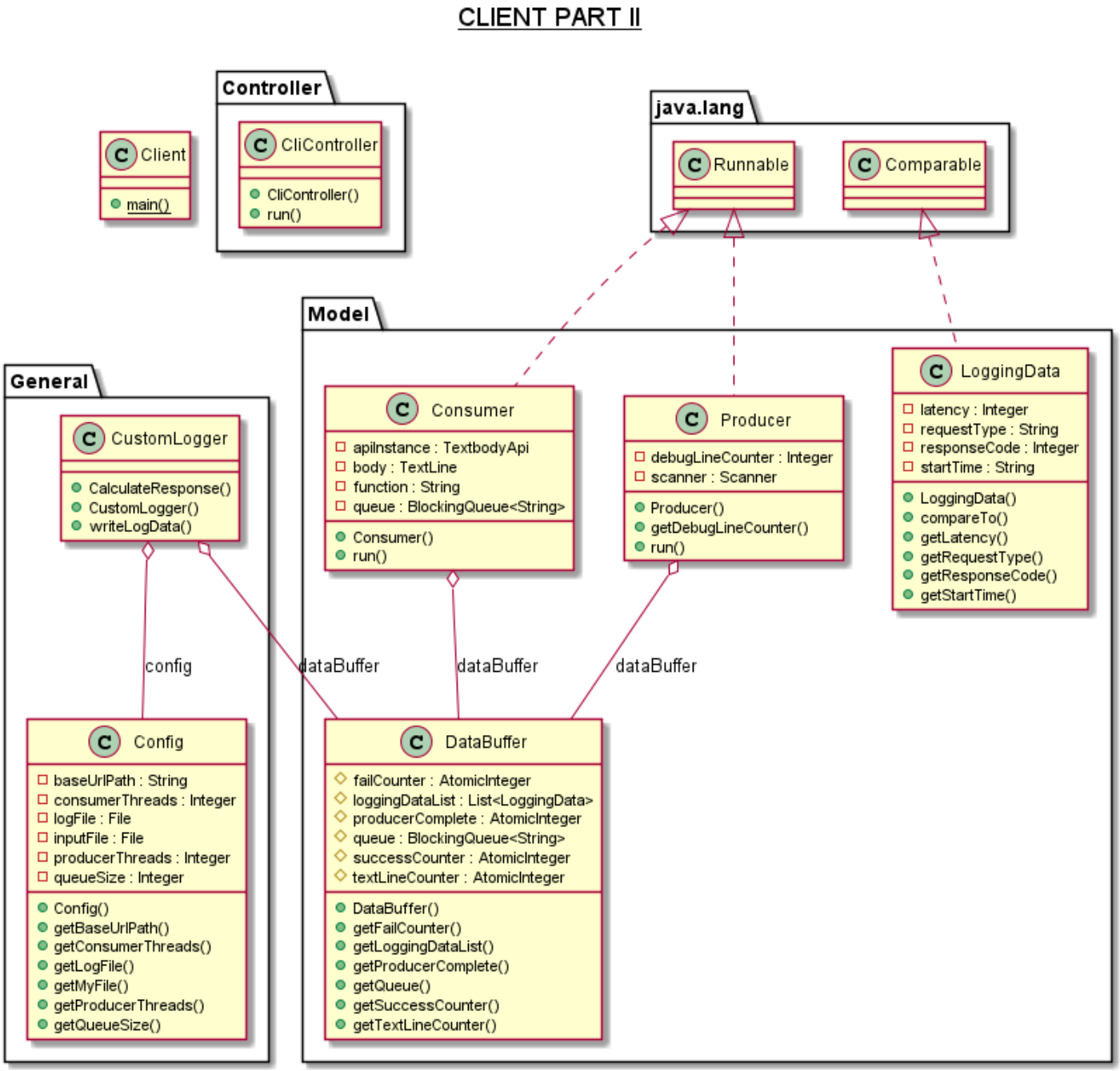
https://gortonator.github.io/bsds-6650/assignments-2021/Assignment-2

Code Repository:

https://github.com/xelese/BSDS/tree/master/Assignment2

Client Description*:*

No changes in the client were made. Made use of https://github.com/gortonator/bsds-6650/blob/master/assignments-2021/bsds-summer-2021-testdata-assignment2.txt this text file for Load testing containing almost 100,000 lines. Client was run locally.



Server Description:

A few changes were made to the server. Firstly, the previous spring boot implementation has been scrapped in favor of Http servlet.

The performance improvement form HTTP servlet was better in comparison to Spring Boot system. On an average if the benchmark for 256 threads was considered at 2500 req/s for Assignment 1, with the wordcount function and RabbitMQ implementation.

a. The Spring boot implementation was showing 1700 req/s at most.
b. The Http servlets was better with about 2300 req/s at minimum.

With some optimization I was able to get the performance equal to Assignment 1 or better.

In the GIT repository you would notice that there are 2 Server implementations. Server and Server2. I have used Server for my final presentation. Server2 is Spring boot test which was considerably slow.

One important note here is regarding RabbitMQ and channel creation. I am currently using Apache Pools2 implementation to create a pool. There were many considerations for different pool sizes for channels. I tested the application with a min of 128 channels maintained with a maximum of 256 channels. In my testing I found out that Tomcat allows you to create only up to 200 channels to RabbitMQ per server.

One small optimization was by creating a local hash map of the request and sending the toString() value of that to the Consumer. That allowed me to return to the client faster and have a better throughput in general.

The server has been created on AWS in two configurations:

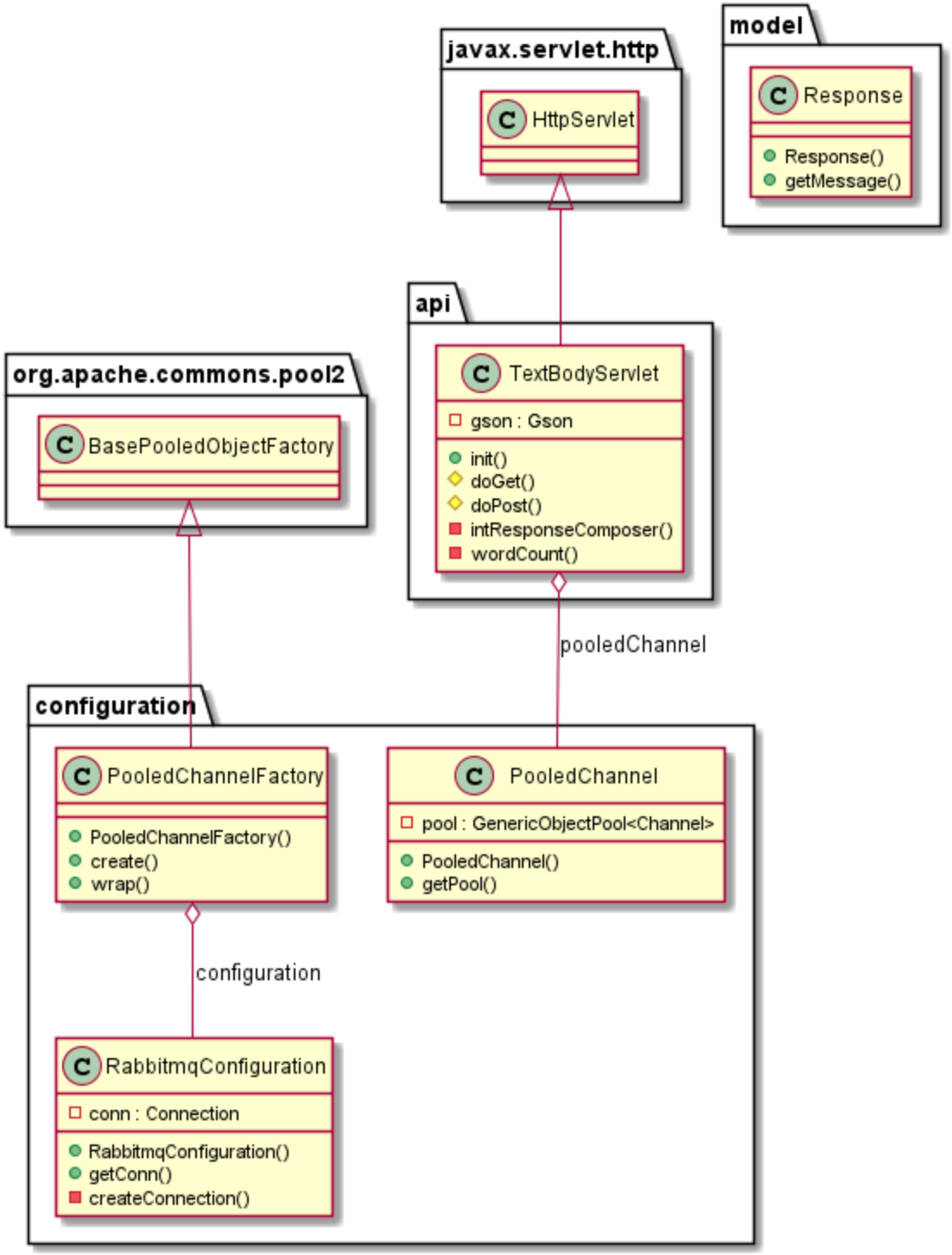
Firstly, a single instance.

- Amazon Linux 2
- T2 Micro
- Tomcat 8.5
- US-EAST-1 (N. Virginia)

Secondly, load balanced via Elastic Beanstalk instance.

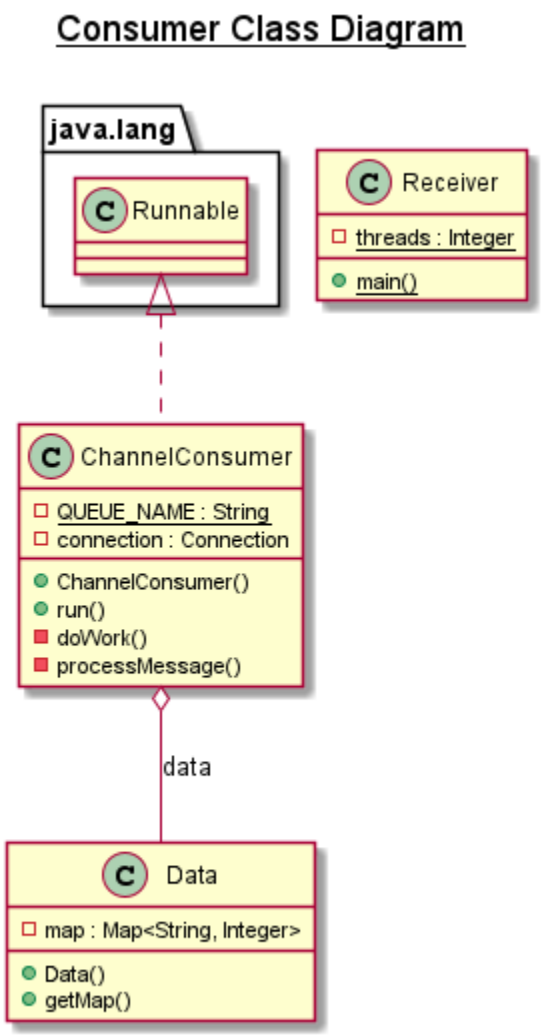
- Amazon Linux 2
- T2 Micro
- Tomcat 8.5
- US-EAST-1 (N. Virginia)
- 4 instances.

HTTP Servlet Class Diagram



Consumer Description:

The consumer is quite simple, it establishes a connection with RabbitMQ and runs pulls data via basicAck() and processes messages into a concurrent hash map. The data here needs to be deconstructed by the consumer before it can be added to the map. In my testing I found 256 threads for the consumer to be ideal number where the consumption always kept up with the client's production. Even at 512 client Threads. One more interesting note is that a new channel was created per thread instead of borrowing it from any pool.



RabbitMQ Description:

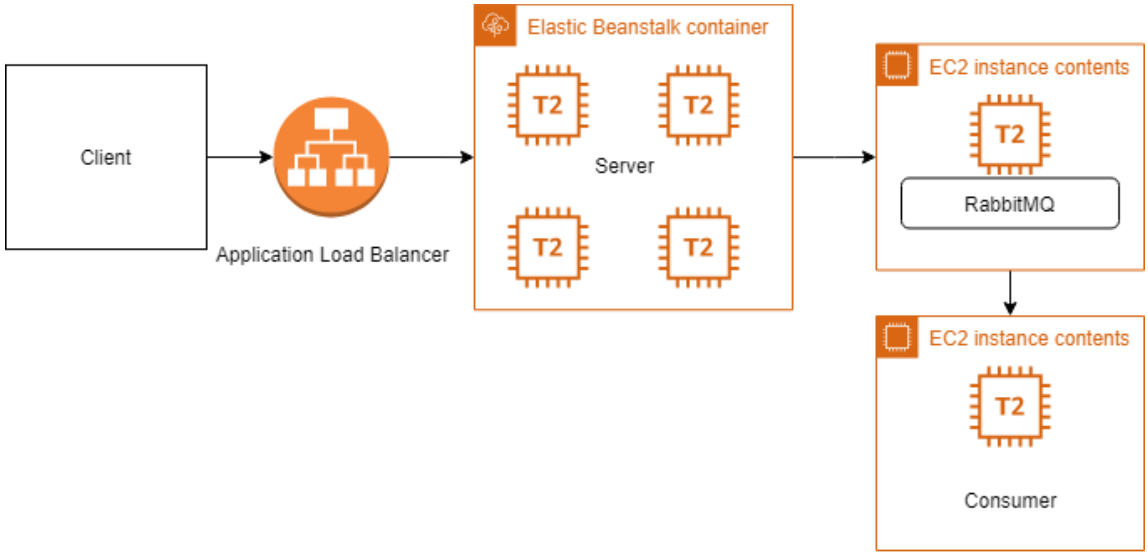
RabbitMQ was hosted on an EC2 instance with the following configuration.

- Amazon Linux 2
- T2 Small
- US-EAST-1 (N. Virginia)
- Elastic IP

I had also assigned an elastic IP to RabbitMQ so that the server and consumer code did not have to be updated every time I stopped the instance.

Overall System Design:

Client connects to an application load balancer which routes the data to one of the instances of server running within the Elastic Beanstalk. Once processed the data is sent to RabbitMQ which is running on a separate EC2 instance with an Elastic IP attached to it. RabbitMQ then pushes the data to Consumer as the Consumer is subscribed to RabbitMQ running on a separate EC2 instance.



Name	I...		S..	A..	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Bsds_server_2	i...	t...	2/	+	us-east-1e	ec2-52-91-133-156.compute-1.amazonaws.com	52.91.133.156	–
Bsds_server_3	i...	t...	2/	+	us-east-1e	ec2-100-25-161-107.compute-1.amazonaws.com	100.25.161.107	–
Bsds_server_4	i...	t...	2/	+	us-east-1e	ec2-54-89-134-153.compute-1.amazonaws.com	54.89.134.153	–
Bsds_rabbitmq	i...	t...	2/	+	us-east-1b	ec2-3-225-35-105.compute-1.amazonaws.com	3.225.35.105	3.225.35.105
Bsds_consumer	i...	t...	2/	+	us-east-1d	ec2-34-207-233-170.compute-1.amazonaws.com	34.207.233.170	–
Bsds_server_1	i...	t...	2/	+	us-east-1c	ec2-3-80-118-152.compute-1.amazonaws.com	3.80.118.152	–

Results:

32 Threads:

No Load Balancer:

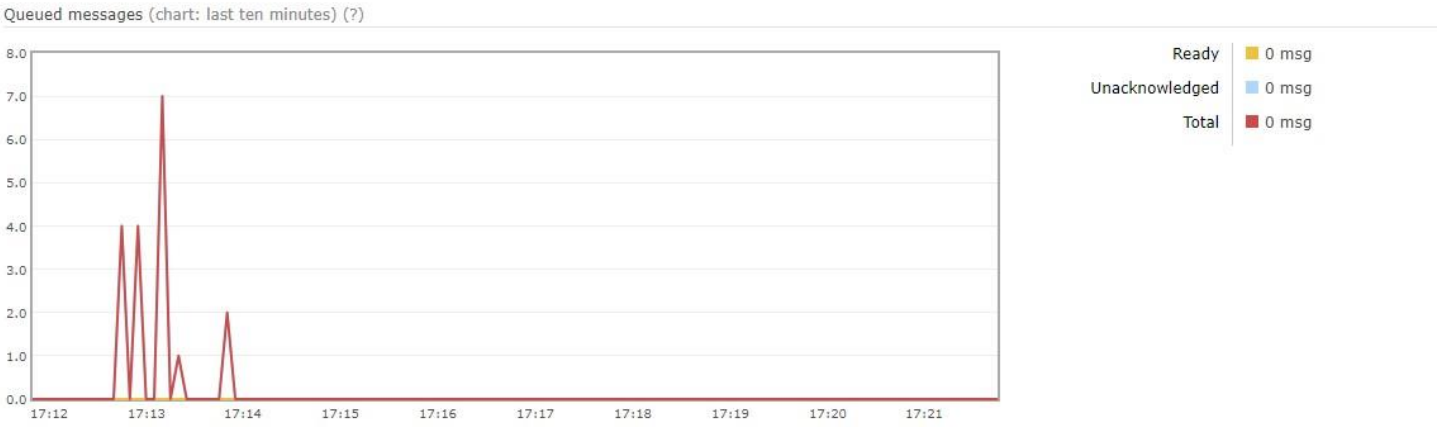
```
All threads terminated.
----------------- CONFIG -------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 32
Total number of client requests consumed: 99300
Log File location: .\log\2021-06-14-17-12-31-log.csv
----------------- OVERALL STATS ------------------
1. Total number of successful requests sent: 99300
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 80.79355 s
4. Throughput: 1229.0585 req/s
----------------- DETAILED STATS -----------------
1. Mean Response Time: 25.517038 ms
1. Median Response Time: 26.5 ms
1. Max Response Time: 23.0 ms
1. 90th Percentile: 25.0 ms
--------------------------------------------------
```

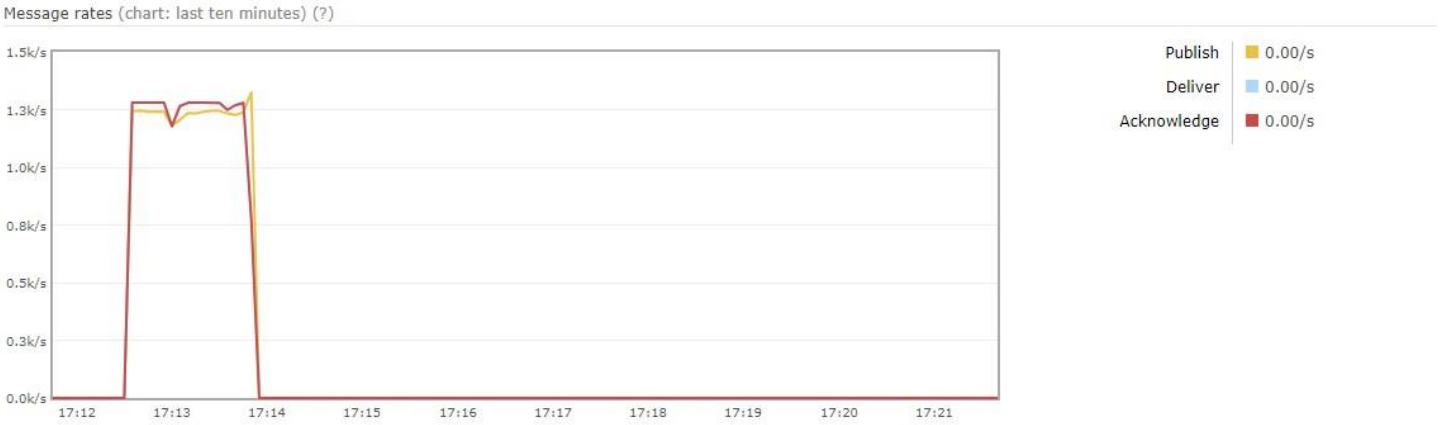
32 Threads:

Load Balancer:

```
All threads terminated.
------------------ CONFIG --------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 32
Total number of client requests consumed: 99300
Log File location: .\log\2021-06-14-11-06-00-log.csv
------------------ OVERALL STATS -------------------
1. Total number of successful requests sent: 99300
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 79.48894 s
4. Throughput: 1249.2305 req/s
------------------ DETAILED STATS ------------------
1. Mean Response Time: 25.09004 ms
1. Median Response Time: 24.0 ms
1. Max Response Time: 34.0 ms
1. 90th Percentile: 22.0 ms
----------------------------------------------------

Process finished with exit code 0
```



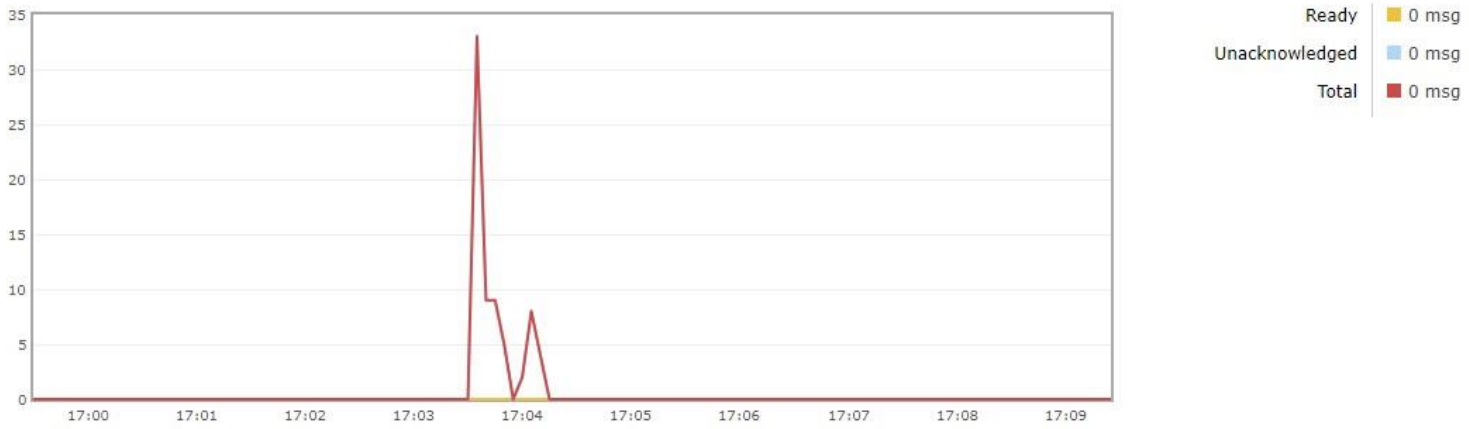


64 Threads:

No Load Balancer:

```
----------------- CONFIG -------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 64
Total number of client requests consumed: 99300
Log File location: .\log\2021-06-14-17-03-27-log.csv
----------------- OVERALL STATS ------------------
1. Total number of successful requests sent: 99300
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 41.727192 s
4. Throughput: 2379.7432 req/s
----------------- DETAILED STATS -----------------
1. Mean Response Time: 26.356375 ms
1. Median Response Time: 28.5 ms
1. Max Response Time: 30.0 ms
1. 90th Percentile: 29.0 ms
--------------------------------------------------
```

Queued messages (chart: last ten minutes) (?)



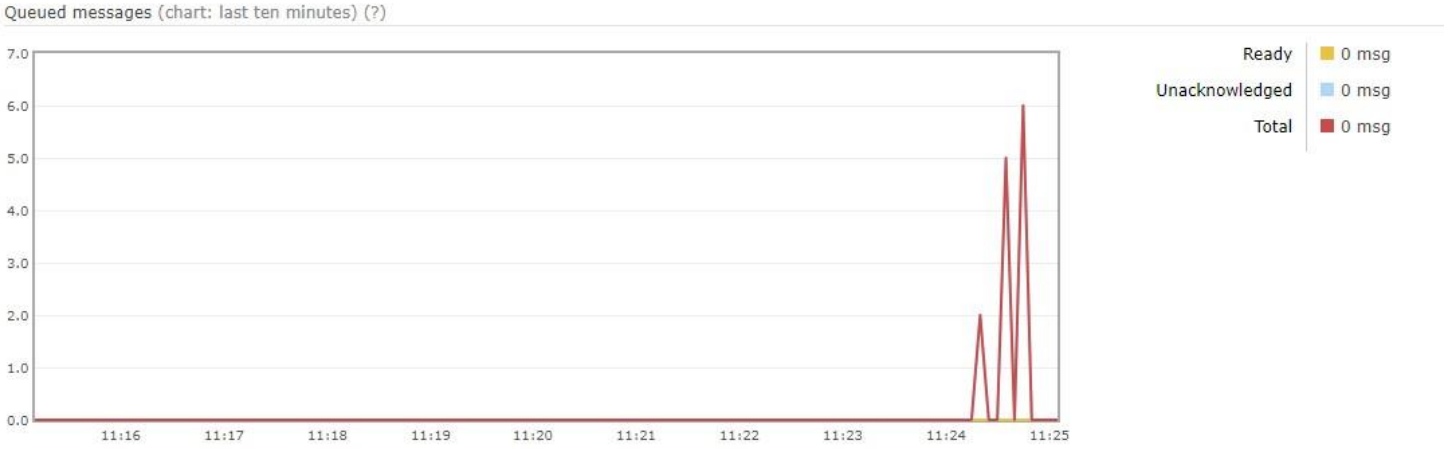
Message rates (chart: last ten minutes) (?)

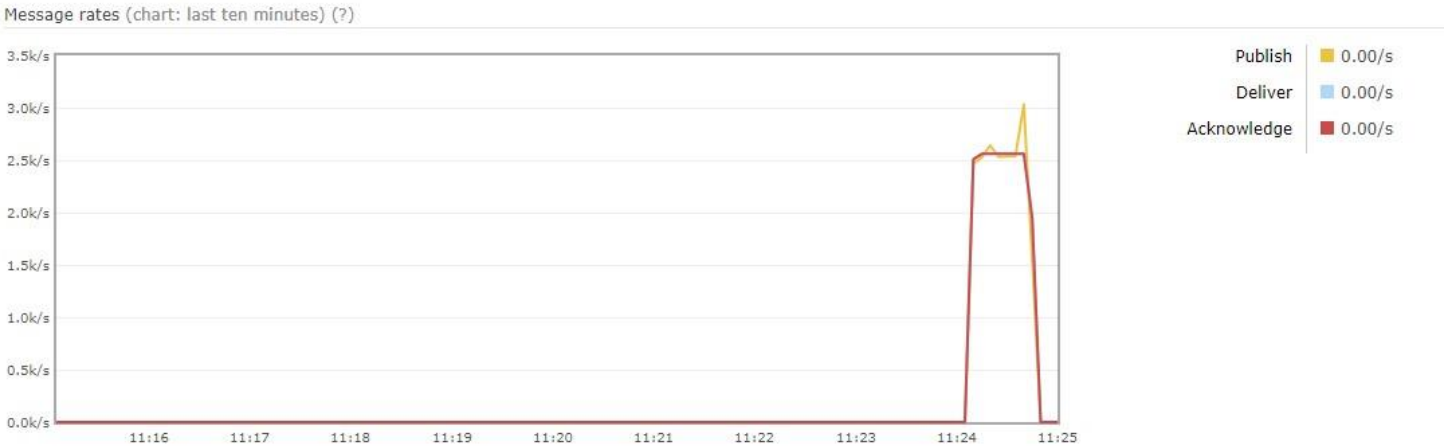
64 Threads:

Load Balancer:

```
All threads terminated.
----------------- CONFIG -------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 64
Total number of client requests consumed: 99300
Log File location: .\log\2021-06-14-11-24-08-log.csv
----------------- OVERALL STATS ------------------
1. Total number of successful requests sent: 99300
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 39.702724 s
4. Throughput: 2501.088 req/s
----------------- DETAILED STATS -----------------
1. Mean Response Time: 25.045881 ms
1. Median Response Time: 23.5 ms
1. Max Response Time: 33.0 ms
1. 90th Percentile: 23.0 ms
--------------------------------------------------

Process finished with exit code 0
```





128 Threads:

No Load Balancer:

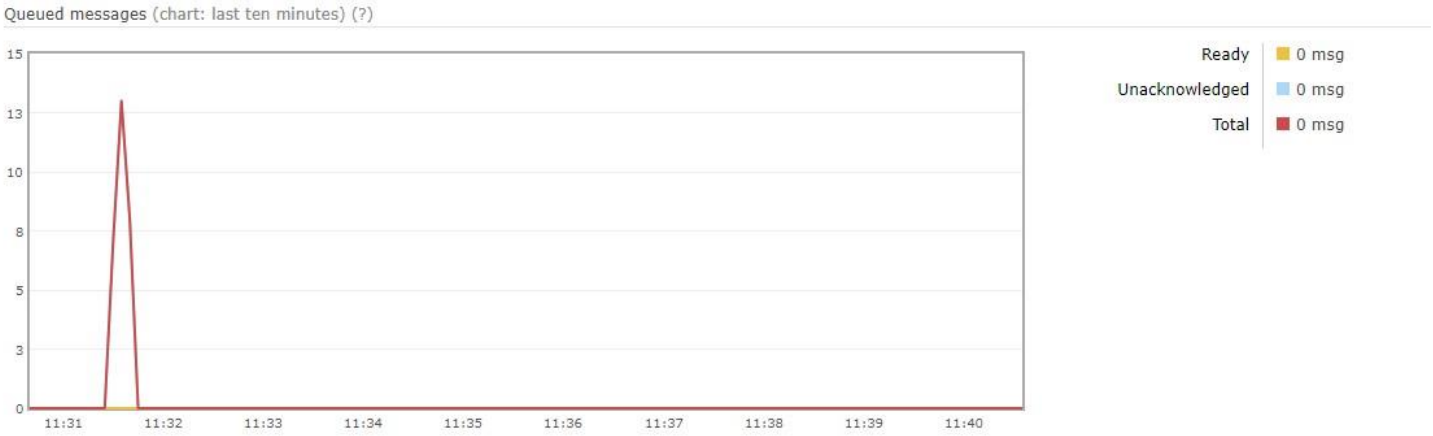
```
All threads terminated.
------------------ CONFIG --------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 128
Total number of client requests consumed: 99300
Log File location: .\log\2021-06-14-17-23-13-log.csv
------------------ OVERALL STATS -------------------
1. Total number of successful requests sent: 99300
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 23.519615 s
4. Throughput: 4222.008 req/s
------------------ DETAILED STATS ------------------
1. Mean Response Time: 29.718933 ms
1. Median Response Time: 30.5 ms
1. Max Response Time: 28.0 ms
1. 90th Percentile: 31.0 ms
----------------------------------------------------
```

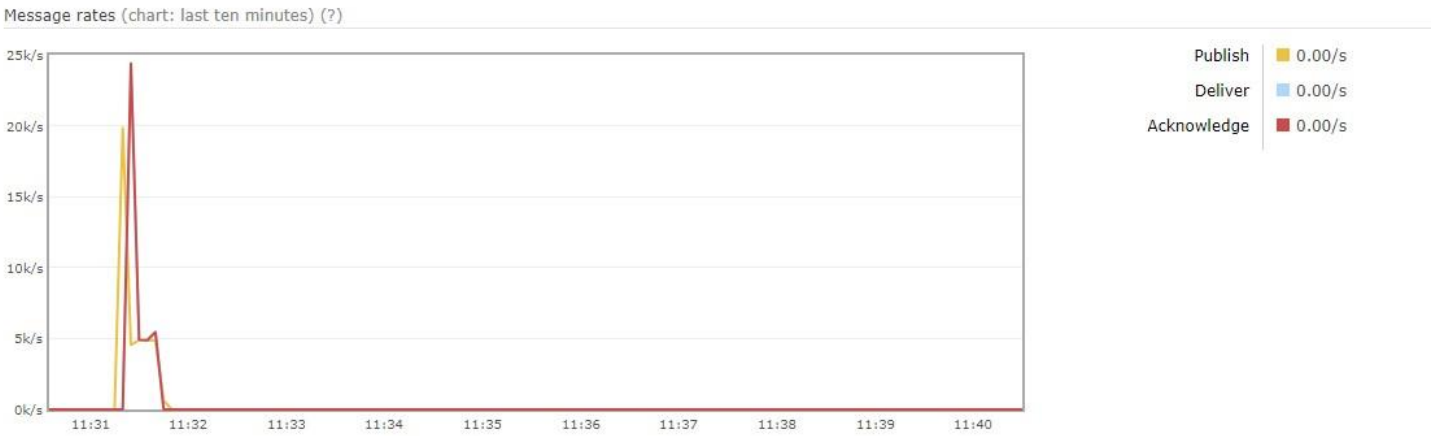
128 Threads:

Load Balancer:

```
All threads terminated.
------------------ CONFIG --------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 128
Total number of client requests consumed: 99300
Log File location: .\log\2021-06-14-11-31-21-log.csv
------------------ OVERALL STATS -------------------
1. Total number of successful requests sent: 99300
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 21.129494 s
4. Throughput: 4699.592 req/s
------------------ DETAILED STATS ------------------
1. Mean Response Time: 26.565428 ms
1. Median Response Time: 25.5 ms
1. Max Response Time: 98.0 ms
1. 90th Percentile: 28.0 ms
----------------------------------------------------

Process finished with exit code 0
```





256 Threads:

No Load Balancer:

```
All threads terminated.
------------------ CONFIG -------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 256
Total number of client requests consumed: 99300
Log File location: .\log\2021-06-14-16-08-08-log.csv
------------------ OVERALL STATS ------------------
1. Total number of successful requests sent: 99300
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 16.04369 s
4. Throughput: 6189.349 req/s
------------------ DETAILED STATS -----------------
1. Mean Response Time: 39.89813 ms
1. Median Response Time: 37.5 ms
1. Max Response Time: 32.0 ms
1. 90th Percentile: 39.0 ms
---------------------------------------------------
```

256 Threads:

Load Balancer:

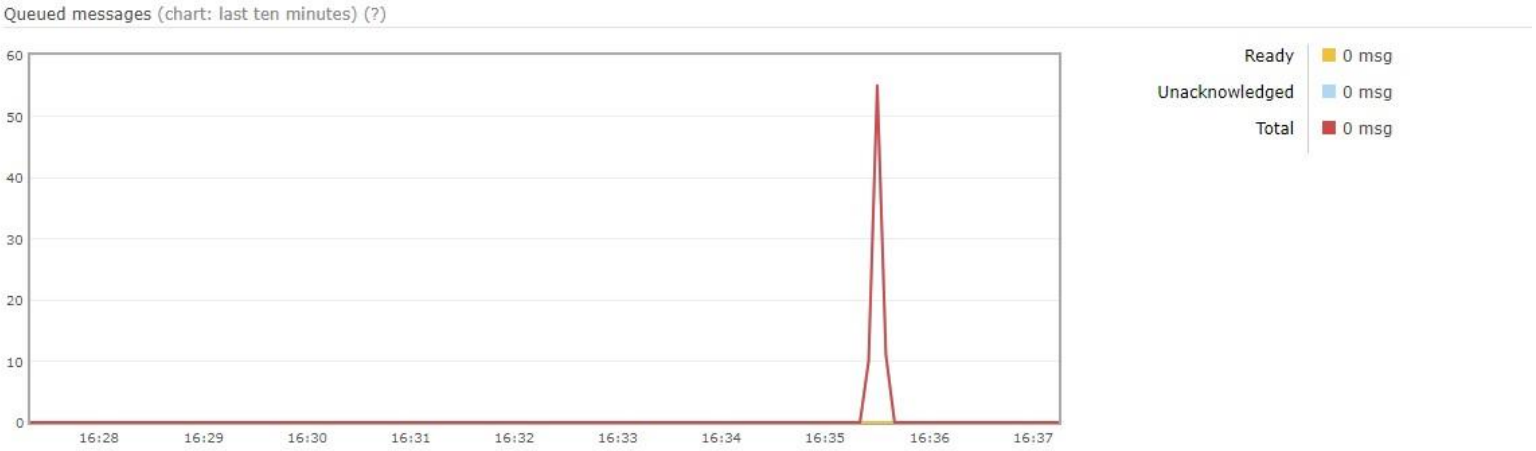
```
All threads terminated.
------------------ CONFIG --------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 256
Total number of client requests consumed: 99300
Log File location: .\log\2021-06-14-12-00-22-log.csv
------------------ OVERALL STATS -------------------
1. Total number of successful requests sent: 99300
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 15.907543 s
4. Throughput: 6242.322 req/s
------------------ DETAILED STATS ------------------
1. Mean Response Time: 36.85151 ms
1. Median Response Time: 28.5 ms
1. Max Response Time: 79.0 ms
1. 90th Percentile: 31.0 ms
----------------------------------------------------

Process finished with exit code 0
```

512 Threads (Bonus):

No Load Balancer:

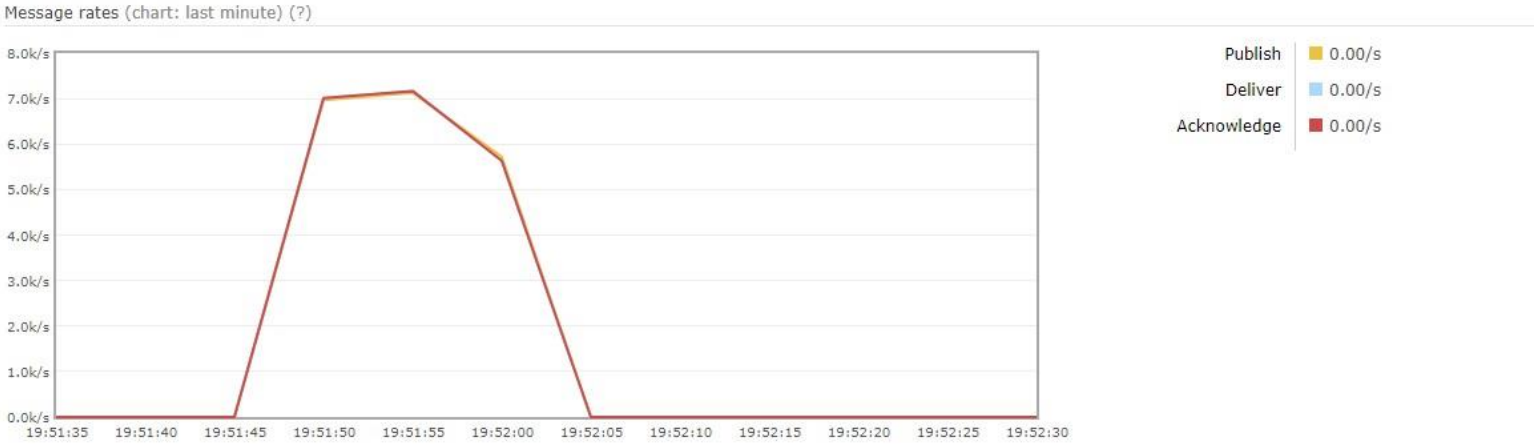
```
All threads terminated.
----------------- CONFIG -------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 512
Total number of client requests consumed: 99300
Log File location: .\log\2021-06-14-16-35-18-log.csv
----------------- OVERALL STATS ------------------
1. Total number of successful requests sent: 99300
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 16.508469 s
4. Throughput: 6015.0947 req/s
----------------- DETAILED STATS -----------------
1. Mean Response Time: 74.247734 ms
1. Median Response Time: 59.5 ms
1. Max Response Time: 553.0 ms
1. 90th Percentile: 120.0 ms
--------------------------------------------------
```



512 Threads (Bonus):

Load Balancer:

```
All threads terminated.
------------------ CONFIG --------------------------
Total number of producer Threads: 1
Total number of consumer Threads: 512
Total number of client requests consumed: 99300
Log File location: .\log\2021-06-16-19-51-45-log.csv
------------------ OVERALL STATS -------------------
1. Total number of successful requests sent: 99300
2. Total number of unsuccessful requests sent: 0
3. Total wall time: 14.685893 s
4. Throughput: 6761.591 req/s
------------------ DETAILED STATS ------------------
1. Mean Response Time: 74.71879 ms
1. Median Response Time: 68.0 ms
1. Max Response Time: 1756.0 ms
1. 90th Percentile: 82.0 ms
----------------------------------------------------
```



Experiments and Analysis:

1. Initially the server was designed using Spring Boot which was extremely slow. I improved it by moving to HTTP servlets instead which gave me a significant boost.

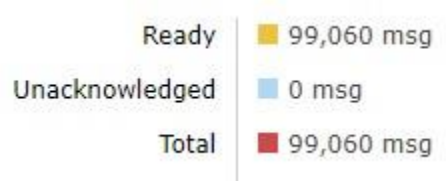
2. The first set up of RMQ was done using direct exchange. Although useful, the direct exchange allows some filtering based on the message's routing key to determine which queue(s) receive(s) the message. There was no necessary reason to have such a method, so I moved to fanout exchange that sends messages to all bound queues.

3. Initially I was sending the message to consumers one at a time that is iterating in the for loop borrowing a channel sending the data and returning the channel back to the pool. This was causing unnecessary overhead on the server and response to client was slow. To improve this, I started sending the toString() of my local map instead which generated fewer requests and shifter the message processing to the consumer improving response times.

4. The first implementation only involved a single instance of the server. It worked well however, if there were any errors or if the server had to stop for processing or any other reason, it would cause a significant delay on the client. To combat that I had implemented the system behind a Load Balancer that made the system highly available. It appears that the load balanced system was only slightly faster than the single instance in returning output to the client, however it brings additional stability to the system by making the server highly available and allowing more requests overall.

5. Based on some assumptions I initially thought that I would need multiple consumers to be running on the system and have it written to a common Elastic Cache to handle the mapping. But in testing I found out that 1 multi-threaded consumer was more than enough to handle heavy loads.

6. In total my server appears to be sending about 99,060 messages.



Based on this and the above info we can calculate the average amount of time items spend I the system using Little's law.

1. L = A x W.
2. Number of items in the system = (the rate items enter and leave the system) x (the average amount of time items spend in the system)
3. W = L / A.

Threads	Rate of items (rounded)		Average amount of time items spent		total_items
	NO LB	LB	NO LB	LB	99060
32	1320	1320	75.04545455	75.04545455	
64	2400	2600	41.275	38.1	
128	4200	5750	23.58571429	17.22782609	
256	7100	7000	13.95211268	14.15142857	
512	7000	7100	14.15142857	13.95211268	

Threads vs Throughput: