

HW 2 Report - Linni Cai

Github URL:

https://github.com/linni-cai-lc/CS6650_Distributed_System/tree/main/hw2

Statistics URL:

https://docs.google.com/spreadsheets/d/1l0VhmU-Bu_0gtQ4B3u6QT-v6oz5fsVwlK6J3LJ9UpsM/edit#gid=0

Server Design:

- ChannelFactory
 - Create a channel to connect to RabbitMQ with authentication on the EC2 instance which runs the RabbitMQ server
- SkierServlet
 - doGet
 - implement GET request
 - distinguish URL parts and obtain parameter information
 - report request status and message
 - doPost
 - implement POST request
 - distinguish URL parts and obtain parameter information
 - obtain request body as a JSON object
 - report request status and message
 - sendDataToQueue
 - pack and publish the JSON string to RabbitMQ's queue

Results:

I run part2's multi-threaded client locally to send POST requests and obtain statistics reports.
I created three EC2 instances:

- Server
 - Linux instance to hold server war, providing skier API functionality
 - has four image copies
 - connect to load balancer
 - send messages to the queue
- RabbitMQ
 - Ubuntu instance to hold RabbitMQ server
 - owns the queue and store messages

- Consumer
 - Linux instance to run consumer jar
 - receive messages from the queue

Instances (3) [Info](#)

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4
<input type="checkbox"/>	Linux (Server)	i-0e965884ba592ab77	Running	t2.micro	2/2 checks passed	No alarms	us-west-2c	ec2-54-200-234-195.us-...	54.200.234.195
<input type="checkbox"/>	Ubuntu (RabbitMQ)	i-066a6081de2958826	Running	t2.micro	2/2 checks passed	No alarms	us-west-2c	ec2-34-221-63-227.us-...	34.221.63.227
<input type="checkbox"/>	Linux (Consumer)	i-051b494b1537cd561	Running	t2.micro	2/2 checks passed	No alarms	us-west-2c	ec2-18-236-237-22.us-...	18.236.237.22

Amazon Machine Images (AMIs) (4) [Info](#)

<input type="checkbox"/>	Name	AMI ID	AMI name	Source	Owner	Visibility	Status
<input type="checkbox"/>	-	ami-00fe905b2b9168613	server-image-2	207705769355/server-image-2	207705769355	Private	Available
<input type="checkbox"/>	-	ami-0a7603b77390023ca	server-image-4	207705769355/server-image-4	207705769355	Private	Available
<input type="checkbox"/>	-	ami-0aa42a4e2ce6ae4f1	server-image-3	207705769355/server-image-3	207705769355	Private	Available
<input type="checkbox"/>	-	ami-0c438da659c07bc43	server-image-1	207705769355/server-image-1	207705769355	Private	Available

<input type="checkbox"/>	Name	DNS name	State	VPC ID	Availability Zones	Type	Created At
<input checked="" type="checkbox"/>	hw2-load-balancer	hw2-load-balancer-1047945...	Active	vpc-0353b78166324872b	us-west-2d, us-west-2a...	application	March 8, 2022 at 11:28:50 A...

hw2-target-group8080

arn:aws:elasticloadbalancing:us-west-2:207705769355:targetgroup/hw2-target-group8080/3a8ec9dfc27ca92a

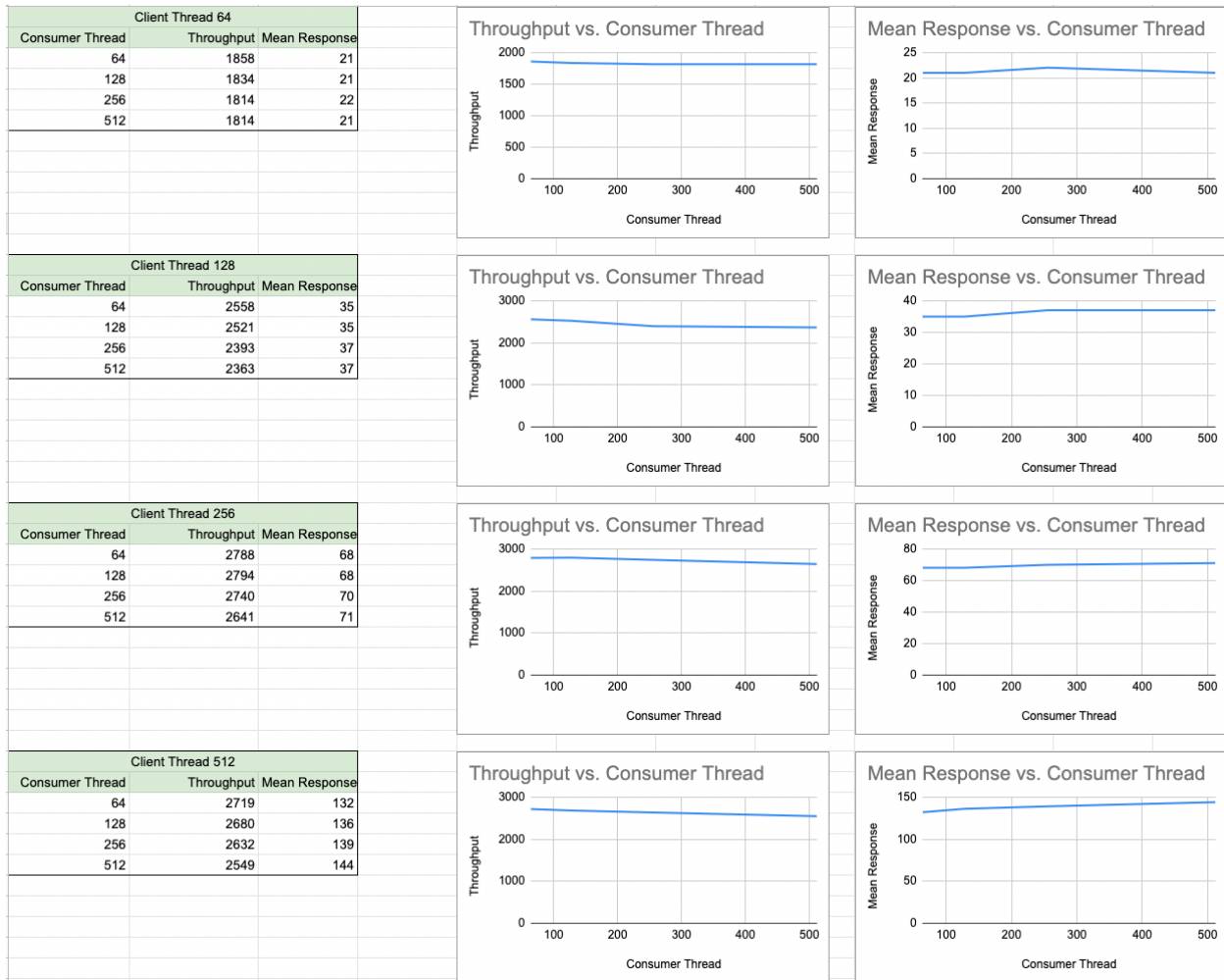
Details

Target type Instance	Protocol : Port HTTP: 8080	Protocol version HTTP1	VPC vpc-0353b78166324872b
IP address type IPv4	Load balancer hw2-load-balancer		

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
1	1	0	0	0	0

I obtained throughput and mean response statistics from part 2's multi-threaded client.

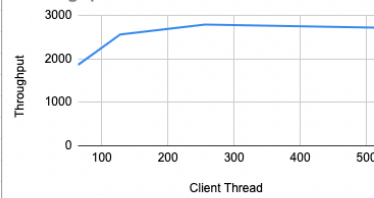
- When I set the constant number of client threads, increase the number of consumer max threads, it doesn't change too much on throughput and mean response, which means the number of consumer max threads might not affect on these outputs.



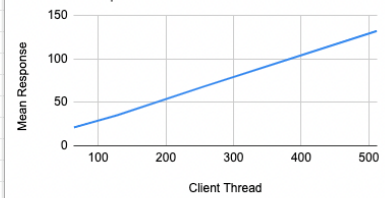
- However, when I set the constant number of consumer threads, increase the number of consumer max threads, there are some effects on throughput and mean response differently.
 - for the throughput, when the number of client threads increases, the throughput increases, however, when it hit a threshold around 256, the throughput becomes stable and not effective anymore.
 - for the mean response, when the number of client threads increases, the mean response time increases, which indicates a positive correlation.

Consumer Thread 64		
Client Thread	Throughput	Mean Response
64	1858	21
128	2558	35
256	2788	68
512	2719	132

Throughput vs. Client Thread

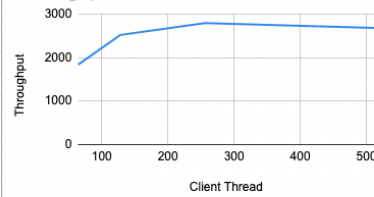


Mean Response vs. Client Thread

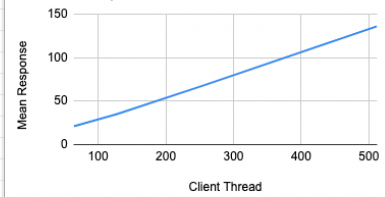


Consumer Thread 128		
Client Thread	Throughput	Mean Response
64	1834	21
128	2521	35
256	2794	68
512	2680	136

Throughput vs. Client Thread

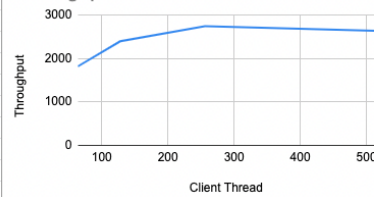


Mean Response vs. Client Thread

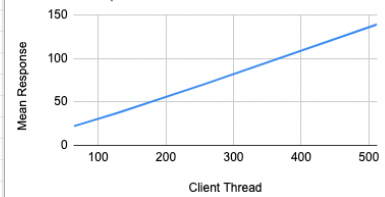


Consumer Thread 256		
Client Thread	Throughput	Mean Response
64	1814	22
128	2393	37
256	2740	70
512	2632	139

Throughput vs. Client Thread

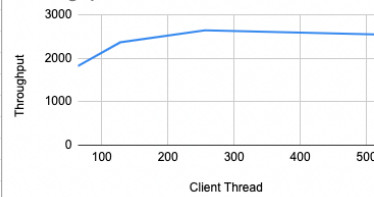


Mean Response vs. Client Thread

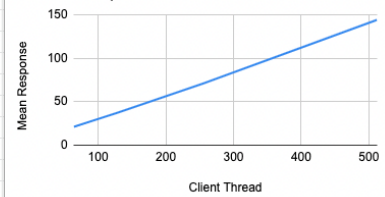


Consumer Thread 512		
Client Thread	Throughput	Mean Response
64	1814	21
128	2363	37
256	2641	71
512	2549	144

Throughput vs. Client Thread



Mean Response vs. Client Thread



With 64 consumer max threads, we have the following RMQ's outputs for the different numbers of client threads, as we can see, the message send/receive rate is around 1, queue size is around 0-15:

- client thread 64

The queue size range is 0 - 10, message rate is around 1.

Queue server_queue

Overview

Queued messages last minute ?

10.0

5.0

0.0

09:04:50

09:05:00

09:05:10

09:05:20

09:05:30

09:05:40

Ready

Unacked

Total

0

3

3

Message rates last minute ?

3.0 k/s

2.0 k/s

1.0 k/s

0.0 k/s

09:04:50

09:05:00

09:05:10

09:05:20

09:05:30

09:05:40

Publish

Deliver (manual ack)

Deliver (auto ack)

2,726/s

2,804/s

0.00/s

Consumer ack

Redelivered

Get (manual ack)

2,804/s

0.00/s

0.00/s

Get (auto ack)

Get (empty)

0.00/s

0.00/s

Details

Features	State	running	Messages ?	Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy	Consumers	64	Message body bytes ?	364 B	0 B	364 B	364 B	0 B	0 B
Operator policy	Consumer capacity ?	100%	Process memory ?	431 kiB					
Effective policy definition									

Runtime Metrics (Advanced)

Reductions (per second) last minute ?

4.0 M/s

3.0 M/s

2.0 M/s

1.0 M/s

0.0 M/s

09:05:00

09:05:10

09:05:20

09:05:30

09:05:40

09:05:50

Reductions

3,058,680/s

Minimum binary virtual heap size in words (min_bin_vheap_size)

46422

Minimum heap size in words (min_heap_size)

233

Maximum generational collections before fullsweep (fullsweep_after)

65535

Number of minor GCs (minor_gcs)

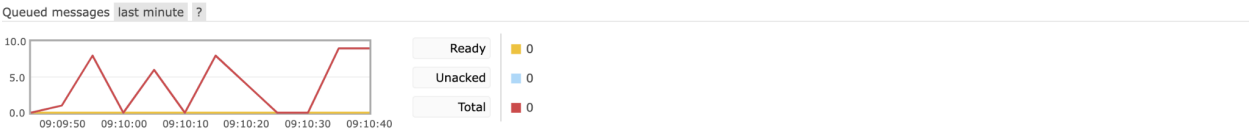
51

- client thread 128

The queue size range is 0 - 10, message rate is around 1.

Queue server_queue

Overview

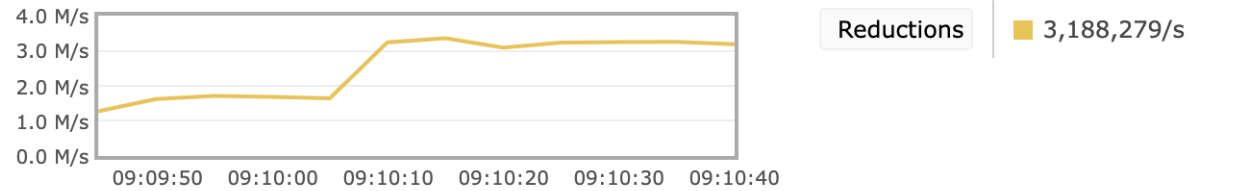


Details

Features	State	running	Messages ?	Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy	Consumers	64	Message body bytes ?	0 B	0 B	0 B	0 B	0 B	0 B
Operator policy	Consumer capacity ?	100%	Process memory ?	431 kiB					
Effective policy definition									

Runtime Metrics (Advanced)

Reductions (per second) last minute ?



Minimum binary virtual heap size in words (min_bin_vheap_size)	46422
Minimum heap size in words (min_heap_size)	233
Maximum generational collections before fullsweep (fullsweep_after)	65535
Number of minor GCs (minor_gcs)	98

- client thread 256

The queue size range is 0 - 15, message rate is around 1.

Queue server_queue

Overview

Queued messages

last minute ?

Ready

Unacked

Total

0

0

0

Message rates

last minute ?

Publish

Deliver (manual ack)

Deliver (auto ack)

3,136/s

3,121/s

0.00/s

Consumer ack

Redelivered

Get (manual ack)

3,121/s

0.00/s

0.00/s

Get (auto ack)

Get (empty)

0.00/s

0.00/s

Details

Features	durable: true	State	running	Messages ?	Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy		Consumers	64	Message body bytes ?	0 B	0 B	0 B	0 B	0 B	0 B
Operator policy		Consumer capacity ?	100%	Process memory ?	431 kiB					
Effective policy definition										

Runtime Metrics (Advanced)

Reductions (per second)

last minute ?

Reductions

3,166,354/s

4.0 M/s

3.0 M/s

2.0 M/s

1.0 M/s

0.0 M/s

09:11:40

09:11:50

09:12:00

09:12:10

09:12:20

09:12:30

Minimum binary virtual heap size in words (min_bin_vheap_size)

46422

Minimum heap size in words (min_heap_size)

233

Maximum generational collections before fullsweep (fullsweep_after)

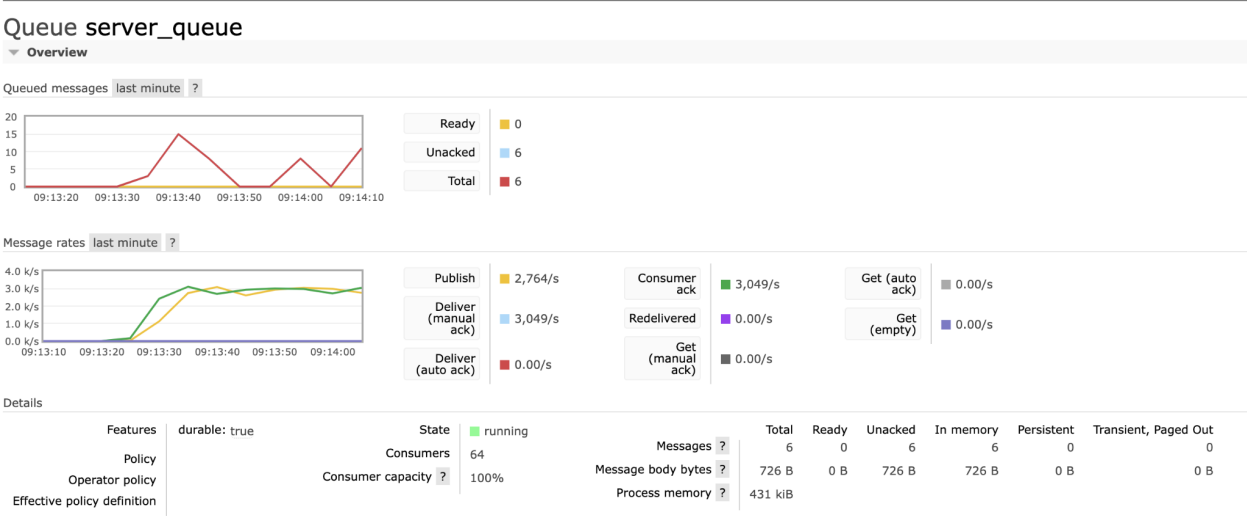
65535

Number of minor GCs (minor_gcs)

29

- client thread 512

The queue size range is 0 - 15, message rate is around 1.



throughput: 2719
p99 (99th percentile) response time: 1604
min response time (milliseconds): 11
max response time (milliseconds): 7885

Consumer - maxThread: 128
mean response time (milliseconds): 136
median response time (milliseconds): 99
throughput: 2680
p99 (99th percentile) response time: 1732
min response time (milliseconds): 12
max response time (milliseconds): 9391

Consumer - maxThread: 256
mean response time (milliseconds): 139
median response time (milliseconds): 100
throughput: 2632
p99 (99th percentile) response time: 1752
min response time (milliseconds): 12
max response time (milliseconds): 9368

Consumer - maxThread: 512
mean response time (milliseconds): 144
median response time (milliseconds): 103
throughput: 2549
p99 (99th percentile) response time: 1887
min response time (milliseconds): 13
max response time (milliseconds): 9401

Client - num_threads: 256

Consumer - maxThread: 64
mean response time (milliseconds): 68
median response time (milliseconds): 64
throughput: 2788
p99 (99th percentile) response time: 278
min response time (milliseconds): 12
max response time (milliseconds): 1292

Consumer - maxThread: 128
mean response time (milliseconds): 68
median response time (milliseconds): 60
throughput: 2794
p99 (99th percentile) response time: 299

min response time (milliseconds): 11
max response time (milliseconds): 1202

Consumer - maxThread: 256
mean response time (milliseconds): 70
median response time (milliseconds): 35
throughput: 2740
p99 (99th percentile) response time: 366
min response time (milliseconds): 12
max response time (milliseconds): 1584

Consumer - maxThread: 512
mean response time (milliseconds): 71
median response time (milliseconds): 29
throughput: 2641
p99 (99th percentile) response time: 404
min response time (milliseconds): 12
max response time (milliseconds): 1417

Client - num_threads: 128

Consumer - maxThread: 64
mean response time (milliseconds): 35
median response time (milliseconds): 36
throughput: 2558
p99 (99th percentile) response time: 77
min response time (milliseconds): 11
max response time (milliseconds): 747

Consumer - maxThread: 128
mean response time (milliseconds): 35
median response time (milliseconds): 36
throughput: 2521
p99 (99th percentile) response time: 75
min response time (milliseconds): 11
max response time (milliseconds): 887

Consumer - maxThread: 256
mean response time (milliseconds): 37
median response time (milliseconds): 37
throughput: 2393
p99 (99th percentile) response time: 86
min response time (milliseconds): 11
max response time (milliseconds): 785

Consumer - maxThread: 512
mean response time (milliseconds): 37
median response time (milliseconds): 39
throughput: 2363
p99 (99th percentile) response time: 79
min response time (milliseconds): 11
max response time (milliseconds): 833

Client - num_threads: 64

Consumer - maxThread: 64
mean response time (milliseconds): 21
median response time (milliseconds): 20
throughput: 1858
p99 (99th percentile) response time: 44
min response time (milliseconds): 11
max response time (milliseconds): 639

Consumer - maxThread: 128
mean response time (milliseconds): 21
median response time (milliseconds): 20
throughput: 1834
p99 (99th percentile) response time: 49
min response time (milliseconds): 10
max response time (milliseconds): 556

Consumer - maxThread: 256
mean response time (milliseconds): 22
median response time (milliseconds): 20
throughput: 1814
p99 (99th percentile) response time: 57
min response time (milliseconds): 11
max response time (milliseconds): 586

Consumer - maxThread: 512
mean response time (milliseconds): 21
median response time (milliseconds): 20
throughput: 1814
p99 (99th percentile) response time: 46
min response time (milliseconds): 11
max response time (milliseconds): 698
