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

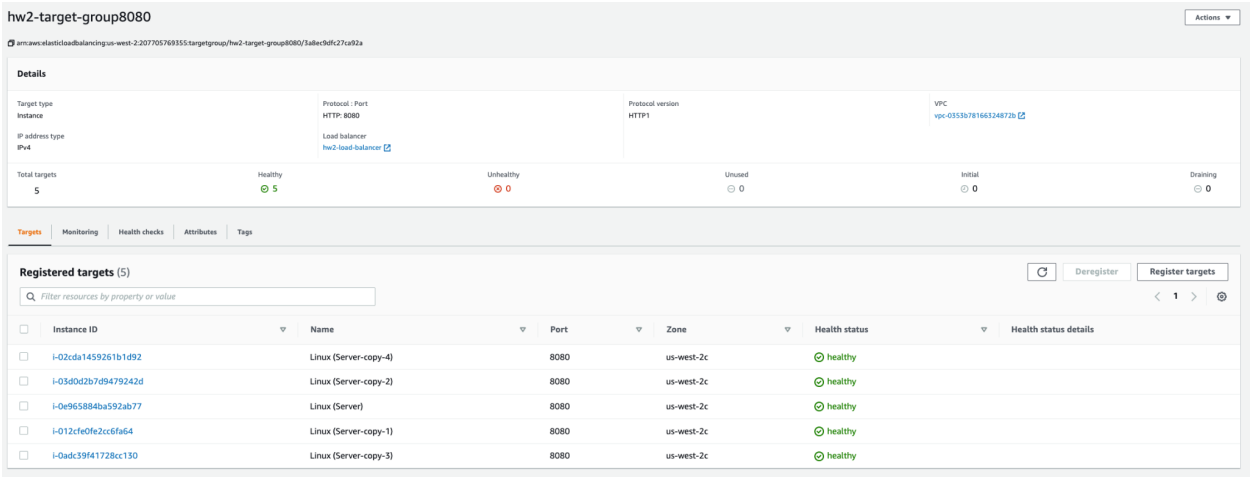
Process:

- created 7 EC2 instance
    - 1 Linux instance running the server
        - provides with the skier API functionality
        - connect to load balancer
        - send messages to the queue
    - 4 Linux instances running the same server image
        - connect to load balancer
    - 1 Linux instance running the consumer
        - receive messages from the queue
    - 1 Ubuntu instance running the RabbitMQ server

- owns the queue and store messages

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
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
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
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
Linux (Server-copy-1)	i-0cf775a7ed326e847	Running	t2.micro	2/2 checks passed	No alarms	us-west-2c	ec2-54-149-57-251.us-...	54.149.57.251
Linux (Server-copy-2)	i-0f2a57255e41b901a	Running	t2.micro	2/2 checks passed	No alarms	us-west-2c	ec2-54-191-193-23.us-...	54.191.193.23
Linux (Server-copy-3)	i-04f5a7613bcb70596	Running	t2.micro	2/2 checks passed	No alarms	us-west-2c	ec2-34-222-35-130.us-...	34.222.35.130
Linux (Server-copy-4)	i-031e41c9429d37b62	Running	t2.micro	2/2 checks passed	No alarms	us-west-2c	ec2-35-88-245-86.us-w...	35.88.245.86

- created a load balancer and a target group, registered the above 5 Linux instances



- run part2's multi-threaded client locally to send POST requests and obtain statistics reports.

Results:

With a load balancer, I set the max thread number as 128, run the client with different numbers of threads. I copied the statistics for reference:

```
consumer max thread = 128
client thread = 512
mean response time (millisecs): 111
median response time (millisecs): 40
throughput: 3284
p99 (99th percentile) response time: 1206
min response time (millisecs): 14
max response time (millisecs): 3611

client thread = 256
mean response time (millisecs): 53
median response time (millisecs): 32
throughput: 3236
p99 (99th percentile) response time: 536
min response time (millisecs): 12
```

```
max response time (millisecs): 3203

client thread = 128
mean response time (millisecs): 28
median response time (millisecs): 25
throughput: 2944
p99 (99th percentile) response time: 76
min response time (millisecs): 11
max response time (millisecs): 918

client thread = 64
mean response time (millisecs): 20
median response time (millisecs): 19
throughput: 1902
p99 (99th percentile) response time: 42
min response time (millisecs): 11
max response time (millisecs): 730
```

The following are RMQ screenshots:

- client thread = 512
- The queue size range is 0 - 120, message rate is send/receive = 3750 / 3698 = 1.01

- client thread = 256
- The queue size range is 0 - 200, message rate is send/receive = 4368 / 4365 = 1.00

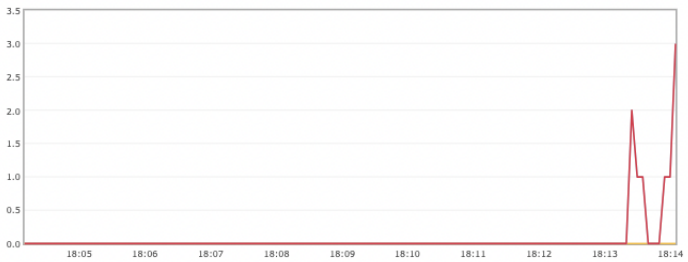
- client thread = 128
- The queue size range is 0 - 50, message rate is send/receive = 3945 / 4157 = 0.95

- client thread = 64
- The queue size range is 0 - 10, message rate is send/receive = 2926 / 3035 = 0.96

Queue server_queue

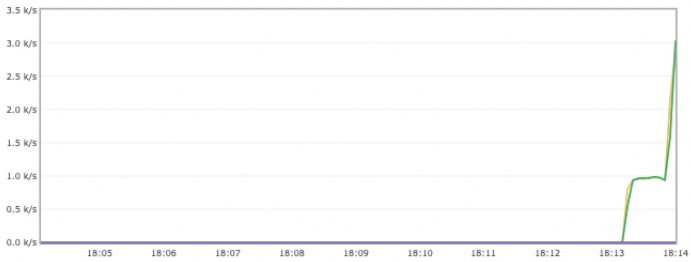
Overview

Queued messages last ten minutes ?



Ready	0
Unacked	5
Total	5

Message rates last ten minutes ?



Publish	2,926/s	Consumer ack	3,035/s	Get (auto ack)	0.00/s
Deliver (manual ack)	3,034/s	Redelivered	0.00/s	Get (empty)	0.00/s
Deliver (auto ack)	0.00/s	Get (manual ack)	0.00/s		

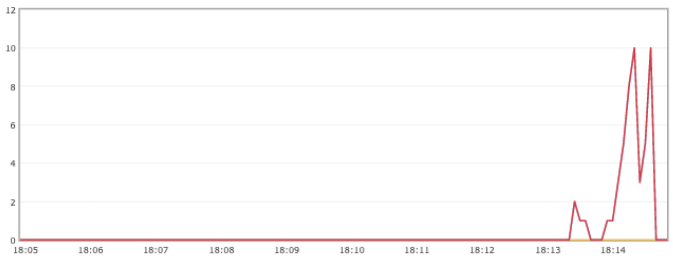
Details

Features	durable: true	State	running		Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy		Consumers	128	Messages ?	5	0	5	5	0	0
Operator policy		Consumer capacity ?	100%	Message body bytes ?	606 B	0 B	606 B	606 B	0 B	0 B
Effective policy definition				Process memory ?	448 kiB					

Queue server_queue

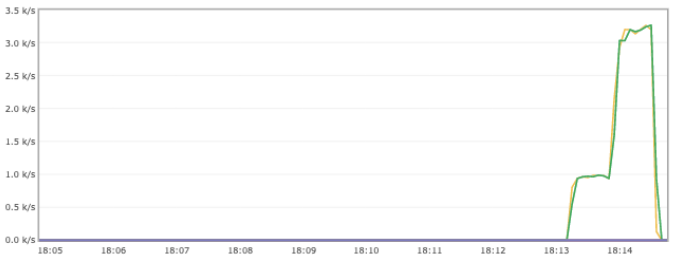
Overview

Queued messages last ten minutes ?



Ready	0
Unacked	0
Total	0

Message rates last ten minutes ?



Publish	0.00/s	Consumer ack	0.00/s	Get (auto ack)	0.00/s
Deliver (manual ack)	0.00/s	Redelivered	0.00/s	Get (empty)	0.00/s
Deliver (auto ack)	0.00/s	Get (manual ack)	0.00/s		

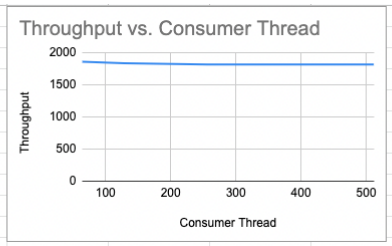
Details

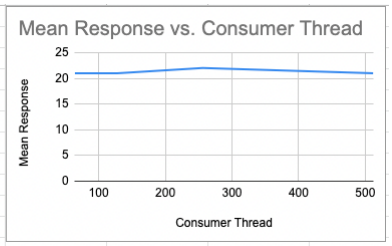
Features	durable: true	State	idle		Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy		Consumers	128	Messages ?	0	0	0	0	0	0
Operator policy		Consumer capacity ?	100%	Message body bytes ?	0 B	0 B	0 B	0 B	0 B	0 B
Effective policy definition				Process memory ?	110 kiB					

Without the load balancer, I also obtained throughput and mean response statistics from part 2's multi-threaded client.

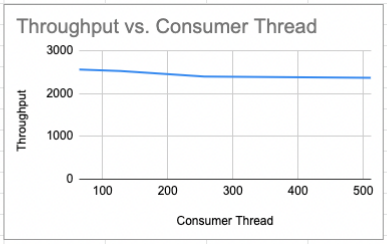
- When I set the constant number of client threads, increase the number of consumer max threads, it doesn't change too much on throughput and mean response, which means the number of consumer max threads might not affect on these outputs.

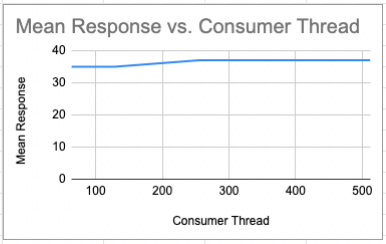
Client Thread 64		
Consumer Thread	Throughput	Mean Response
64	1858	21
128	1834	21
256	1814	22
512	1814	21



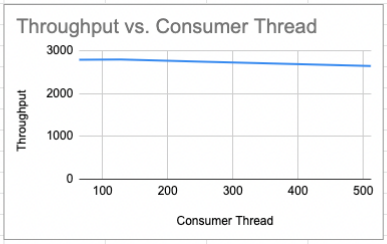


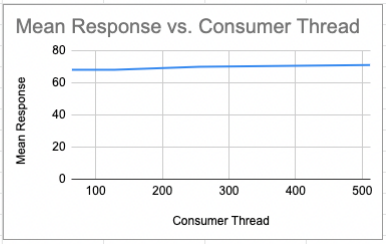
Client Thread 128		
Consumer Thread	Throughput	Mean Response
64	2558	35
128	2521	35
256	2393	37
512	2363	37



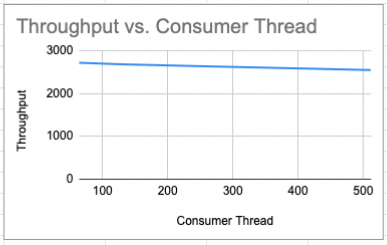


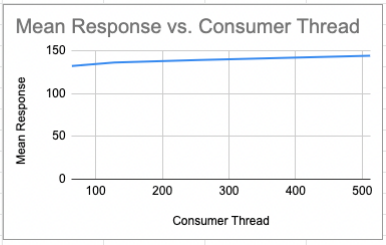
Client Thread 256		
Consumer Thread	Throughput	Mean Response
64	2788	68
128	2794	68
256	2740	70
512	2641	71





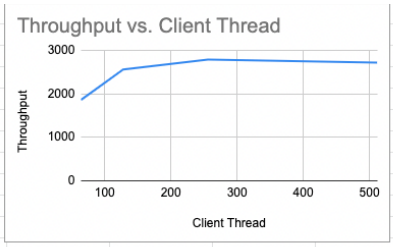
Client Thread 512		
Consumer Thread	Throughput	Mean Response
64	2719	132
128	2680	136
256	2632	139
512	2549	144

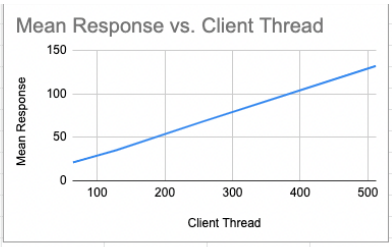




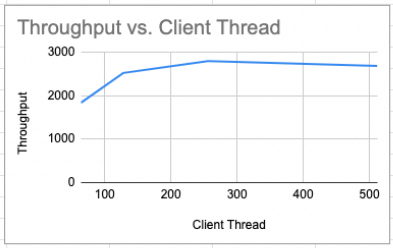
- However, when I set the constant number of consumer threads, increase the number of consumer max threads, there are some effects on throughput and mean response differently.
    - for the throughput, when the number of client threads increases, the throughput increases, however, when it hit a threshold 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

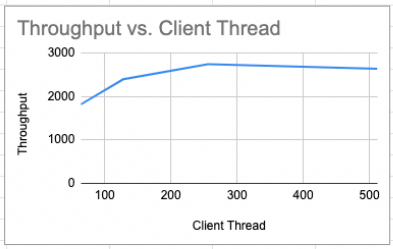


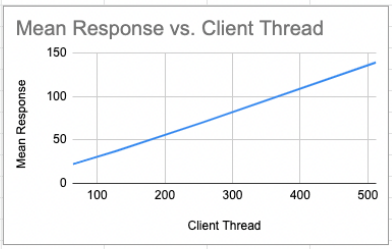


Consumer Thread 128		
Client Thread	Throughput	Mean Response
64	1834	21
128	2521	35
256	2794	68
512	2680	136

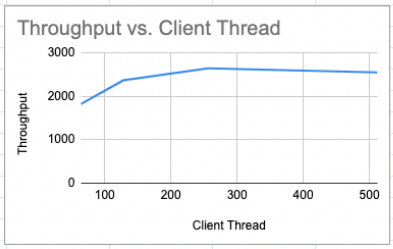


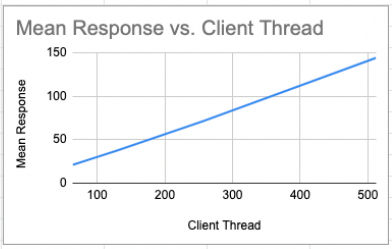
Consumer Thread 256		
Client Thread	Throughput	Mean Response
64	1814	22
128	2393	37
256	2740	70
512	2632	139





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





Without the load balancer, the following statistics are running part2's multi-threaded client with the argument as following includes the various num_threads: 64, 128, 256, 512

```
--num_skiers 20000 --num_lifts 40 --ip_address LoadBalancerDNS:8080
-------------------------------------------------------------
Client - num_threads: 512

Consumer - maxThread: 64
Without Load Balancer:
mean response time (millisecs): 132
median response time (millisecs): 94
throughput: 2719
p99 (99th percentile) response time: 1604
min response time (millisecs): 11
max response time (millisecs): 7885
```

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

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

Consumer - maxThread: 512
mean response time (millisecs): 144
median response time (millisecs): 103
throughput: 2549
p99 (99th percentile) response time: 1887
min response time (millisecs): 13
max response time (millisecs): 9401
-----------------------------------------------------
Client - num_threads: 256

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

Consumer - maxThread: 128
mean response time (millisecs): 68
median response time (millisecs): 60
throughput: 2794
p99 (99th percentile) response time: 299
min response time (millisecs): 11
max response time (millisecs): 1202

Consumer - maxThread: 256
```

```
mean response time (millisecs): 70
median response time (millisecs): 35
throughput: 2740
p99 (99th percentile) response time: 366
min response time (millisecs): 12
max response time (millisecs): 1584

Consumer - maxThread: 512
mean response time (millisecs): 71
median response time (millisecs): 29
throughput: 2641
p99 (99th percentile) response time: 404
min response time (millisecs): 12
max response time (millisecs): 1417
------------------------------------------------------------
Client - num_threads: 128

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

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

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

Consumer - maxThread: 512
mean response time (millisecs): 37
```

```
median response time (millisecs): 39
throughput: 2363
p99 (99th percentile) response time: 79
min response time (millisecs): 11
max response time (millisecs): 833
------------------------------------------------------
Client - num_threads: 64

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

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


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

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