

Sorry for late submission since I requested a one week deadline extension from Prof. Ian due to personal emergency.

Assignment 2

CS 6650, Fall 2023, Ximing Liang

[Q1](#)

[Q2](#)

[Q3](#)

[Q4](#)

[Q5](#)

[Optimization 1: DB connection pooling](#)

[Optimization 2: Co-locate Availability Zone](#)

[Optimization 3: Run the Client from an EC2 Instance](#)

[Final Result](#)

[Q3, Q4, Q5 Comparison](#)

[Appendix](#)

[Target Group](#)

[Load Balancer](#)

[Auto Scaling Group](#)

[RDS](#)

Q1

GitHub Repository URL

<https://github.com/ximing0116/CS6650-Assignment2>

Q2

A short description of your data model (5 points) - Please state size of image used if not using the stock image, and also Database/File storage solution.

Below is my data model:

```
CREATE TABLE albums (  
  albumID serial PRIMARY KEY,  
  artist VARCHAR(255) NOT NULL,
```

```

title VARCHAR(255) NOT NULL,
year VARCHAR(4) NOT NULL
);
CREATE TABLE images (
imageID serial PRIMARY KEY,
albumID INT REFERENCES albums(albumID),
imageData BYTEA NOT NULL
);

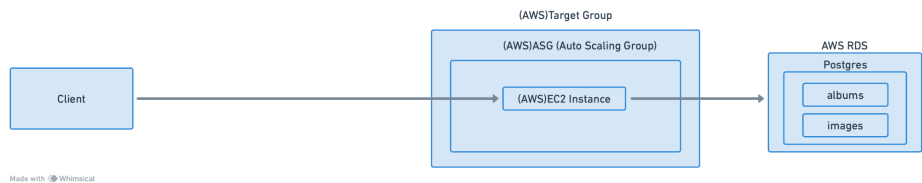
```

There are a few benefits:

- Images are separated from albums profile information, which follows the principle of normalization to avoid storing duplicate images for multiple albums sharing the same image.
- Provides flexibility to potentially store albums and images tables in different hardwares due to their distinct storage and access pattern. For example, the albums table may be updated more frequently than the images table, therefore storing them together incurs unnecessary overhead.

Q3

Output windows for the 3 client configuration tests run against a single server/DB (5 points)



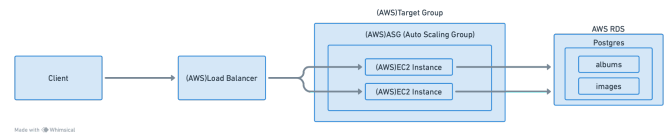
Parameters	Screenshot	Throughput	Failure	Post Stats	Get Stats
------------	------------	------------	---------	------------	-----------

10, 30, 2	<pre>===== Execution started at: 1699825938392 Execution ended at: 1699826433826 ===== Total execution time (sec): 495 Total throughput (reqs/sec): 1213 ===== Total successful total: 601010 Total failed non-200 total: 0 Total failed with exception total: 0 ===== POST Stats: Mean response time (ms): 224.59112029599012 Median response time (ms): 142.0 p99 response time (ms): 1064 Min response time (ms): 21 Max response time (ms): 5062 ===== GET Stats: Mean response time (ms): 211.580146179402 Median response time (ms): 130.0 p99 response time (ms): 1051 Min response time (ms): 16 Max response time (ms): 5042 =====</pre>	1213	0	Mean:224 Median:142 P99:1064 Min:21 Max:5062	Mean:211 Median:130 P99:1051 Min:16 Max:5042
10, 20, 2	<pre>===== Execution started at: 1699825569025 Execution ended at: 1699825901782 ===== Total execution time (sec): 332 Total throughput (reqs/sec): 1205 ===== Total successful total: 401010 Total failed non-200 total: 0 Total failed with exception total: 0 ===== POST Stats: Mean response time (ms): 152.3910004499775 Median response time (ms): 130.0 p99 response time (ms): 590 Min response time (ms): 20 Max response time (ms): 4007 ===== GET Stats: Mean response time (ms): 143.08154228855722 Median response time (ms): 122.0 p99 response time (ms): 493 Min response time (ms): 17 Max response time (ms): 2395 =====</pre>	1205	0	Mean:152 Median:130 P99:590 Min:20 Max:4007	Mean:143 Median:122 P99:493 Min:17 Max:2395

10, 10, 2	<pre>===== Execution started at: 1699825359441 Execution ended at: 1699825515252 ===== Total execution time (sec): 164 Total throughput (reqs/sec): 1219 ===== Total successful total: 201810 Total failed non-200 total: 0 Total failed with exception total: 0 ===== POST Stats: Mean response time (ms): 77.81175882411759 Median response time (ms): 67.0 p99 response time (ms): 278 Min response time (ms): 21 Max response time (ms): 2046 ===== GET Stats: Mean response time (ms): 68.93534653465346 Median response time (ms): 60.0 p99 response time (ms): 301 Min response time (ms): 17 Max response time (ms): 1225 =====</pre>	1219	0	Mean:77 Median:67 P99:278 Min:21 Max:2046	Mean:68 Median:60 P99:301 Min:17 Max:1225
-----------	---	------	---	---	---

Q4

Output windows for the 3 client configuration tests run against a two load balanced servers/DB (15 points)



Parameters	Screenshot Please note the use of ALB endpoint	Throughput	Failure	Post Stats	Get Stats
------------	---	------------	---------	------------	-----------

10, 30, 2	<pre>pool-2-thread-289 - POST request to http://6666-1-187656531.us-east-2.elb.amazonaws.com:8080/190709/AlbumStore/1.0.0/albums succeeded! AlbumId: 1320106 pool-2-thread-289 - GET request to http://6666-1-187656531.us-east-2.elb.amazonaws.com:8080/190709/AlbumStore/1.0.0/albums/1320106 succeeded! Response: {"artist":"xixingli","title":"dora","year":"2014"} ===== Execution started at: 1699827392603 Execution ended at: 1699827890864 ===== Total execution time (sec): 497 Total throughput (req/sec): 1288 ===== Total successful total: 4881910 Total failed non-200 total: 0 Total failed with exception total: 0 ===== POST Stats: Mean response time (ms): 234.214799566832 Median response time (ms): 146.0 p99 response time (ms): 1322 Min response time (ms): 20 Max response time (ms): 24169 ===== GET Stats: Mean response time (ms): 203.43592325581396 Median response time (ms): 141.0 p99 response time (ms): 1187 Min response time (ms): 17 Max response time (ms): 8345</pre>		0	Mean:234 Median:160 P99:1322 Min:22 Max:24169	Mean:203 Median:141 P99:1187 Min:17 Max:83451208
10, 20, 2	<pre>pool-2-thread-186 - POST request to http://6666-1-187656531.us-east-2.elb.amazonaws.com:8080/190709/AlbumStore/1.0.0/albums succeeded! AlbumId: 1320094 pool-2-thread-186 - GET request to http://6666-1-187656531.us-east-2.elb.amazonaws.com:8080/190709/AlbumStore/1.0.0/albums/1320094 succeeded! Response: {"artist":"xixingli","title":"dora","year":"2014"} ===== Execution started at: 1699827393020 Execution ended at: 16998277558874 ===== Total execution time (sec): 339 Total throughput (req/sec): 1184 ===== Total successful total: 488109 Total failed non-200 total: 0 Total failed with exception total: 0 ===== POST Stats: Mean response time (ms): 163.32962851857408 Median response time (ms): 101.0 p99 response time (ms): 1185 Min response time (ms): 20 Max response time (ms): 18214 ===== GET Stats: Mean response time (ms): 135.5967918477613 Median response time (ms): 86.0 p99 response time (ms): 886 Min response time (ms): 17 Max response time (ms): 4702</pre>	1180	0	Mean:163 Median:101 P99:1105 Min:20 Max:18314	Mean:135 Median:86 P99:886 Min:17 Max:4702

10, 10, 2	<pre>post-2-thread-07 - POST request to http://e6d8-1-1876766533.us-west-2.elb.amazonaws.com:8080/album?albumId=20084 succeeded! AlbumId: 20084 post-2-thread-07 - GET request to http://e6d8-1-1876766533.us-west-2.elb.amazonaws.com:8080/album?albumId=20084 succeeded! Response: {"artist":"xixingli","title":"dora","year":"2014"} ===== Execution started at: 1699826793748 Execution ended at: 1699826964219 ===== Total execution time (sec): 170 Total throughput (reqs/sec): 1179 ===== Total successful total: 200108 Total failed non-200 total: 0 Total failed with exception total: 0 ===== POST Stats: Mean response time (ms): 85.9891110878912 Median response time (ms): 67.0 p99 response time (ms): 415 Min response time (ms): 21 Max response time (ms): 3697 ===== GET Stats: Mean response time (ms): 67.87658115841585 Median response time (ms): 58.8 p99 response time (ms): 333 Min response time (ms): 17 Max response time (ms): 3281</pre>	1179	0	Mean:83 Median:67 P99:415 Min:31 Max:3697	Mean:67 Median:58 P99:333 Min:17 Max:3281
-----------	---	------	---	---	---

Q5

Output window for optimized server configuration for client with 30 Thread Groups. Briefly describe what configuration changes you made and what % throughput improvement you achieved (15 points)

Optimization 1: DB connection pooling

Incorperated <https://github.com/brettwooldridge/HikariCP> in my code to pool the DB connections, instead of creating a new connection per request. Results reported in Q3 and Q4 already used this optimization.

Without this optimization, many non-200 were returned due to connection errors to DB. This optimization is required to complete the tests without error.

Optimization 2: Co-locate Availability Zone

Noticed EC2 instances, DB instances and Load balancers are created in different availability zones, which can cause network overhead since availability zones are geographically spread.

Recreated load-balancer and RDS to match the availability zone of the EC2 instances.

EC2 in us-west-2a

	i-0d8c86d9b018814a		t2.micro		3/1 checks passed	No alarms	+	us-west-2a	ec2-53-38-207-38.us-we...	52.38.207.39
	i-032aa50a2a2a2aa65		t2.micro		3/2 checks passed	No alarms	+	us-west-2a	ec2-55-92-243-161.us-...	55.92.243.161

Load-balancer in us-west-2a

Load balancer type Network	Status Active	VPC vpc-0387714eef1306a7	IP address type IPv4
Scheme Internet-facing	Hosted zone Z18D5FSROUN65G	Availability Zones subnet-0f25cfa36f691704 us-west-2a (usw2-az2)	Date created November 12, 2023, 16:00 (UTC-08:00)

DB in us-west-2s

database-2	Available	Instance	PostgreSQL	us-west-2a	db.t4g.micro	2 Actions	<div><div></div></div> 2.94%	<div><div></div></div> 0.00 sessions	none	vpc-0387714eef1306a7	No
----------------------------	------------------------	----------	------------	------------	--------------	-----------	------------------------------	--------------------------------------	------	----------------------	----

Optimization 3: Run the Client from an EC2 Instance

After checking the CPU/Memory/Network metrics of the ec, db and load-balancer instances, I was not able to find a bottleneck, in particular I observed that all CPU usages are below 50%, no memory pressure and network PPS/Throughput is way under limit.

I started to suspect that the bottleneck is actually the client running on my laptop, for the following reasons:

1. My laptop is far-away (a few hundred miles) from the us-west data center. The networking latency between aws data center and my laptop can be high.
2. The uplink of home networking is slow and can limit throughput.

<input checked="" type="checkbox"/>	client	i-0f5013b0f6593caac	Running	t2.large	2/2 checks passed	No alarms	+	us-west-2b
<input type="checkbox"/>		i-0a8cc86adb92d814a	Running	t2.micro	2/2 checks passed	No alarms	+	us-west-2a
<input type="checkbox"/>		i-052aa30a2a52aaa65	Running	t2.micro	2/2 checks passed	No alarms	+	us-west-2a

Final Result

- t2.large instance as my client
- 10 connections pool to Postgres
- All services located in the same availability zone

This shows how I ran the client code in an EC2 instance.

```
[ec2-user@ip-172-31-17-246 ~]$ java -cp "/jackson-databind-2.13.0.jar:/jackson-core-2.13.0.jar" SimpleClient.java RequestMetrics.java Metrics.java: java -cp "/jackson-annotations-2.13.0.jar:/jackson-databind-2.13.0.jar:/jackson-core-2.13.0.jar" SimpleClient 10 30 2 http://6650-1-187696633.us-west-2.elb.amazonaws.com:8080/1000XON/AlbumStore/1.0.0/albums

Execution started at: 1699841548395
Execution ended at: 1699841730479
=====
Total execution time (sec): 182
Total throughput (reqs/sec): 3300
=====
Total successful total: 601010
Total failed non-200 total: 0
Total failed with exception total: 0
=====
POST Stats:
Mean response time (ms): 65.75990800306657
Median response time (ms): 40.0
p99 response time (ms): 460
Min response time (ms): 3
Max response time (ms): 7111
=====
GET Stats:
Mean response time (ms): 58.194215946843855
Median response time (ms): 32.0
p99 response time (ms): 442
Min response time (ms): 1
Max response time (ms): 8519
```

Q3, Q4, Q5 Comparison

Comparison of performance for the (10 30 2) Test across Q3, Q4 and Q5. All latency in millisecond.

	Throughput	Error	Mean	Median	P99	Min	Max
Q3	1213 rps/s	0	Post:224 Get:211	Post:142 Get:130	Post:1064 Get:1051	Post:21 Get:16	Post:5062 Get:5042
Q4	1208 rps/s	0	Post:234 Get:203	Post:160 Get:141	Post:1322 Get:1187	Post:22 Get:17	Post:24169 Get:83451
Q5	3300 rps/s	0	Post:65 Get:58	Post:40 Get:32	Post:460 Get:442	Post:3 Get:1	Post:7111 Get:8519

Appendix

Target Group

Associated with 2 healthy EC2 instances

EC2 > Target groups > 6650-1

6650-1

Details

arn:aws:elasticloadbalancing:us-west-2:241585107714:targetgroup/6650-1/88b5106a461282

Target type

Instance

IP address type

IPv4

Protocol : Port

HTTP:8080

Load balancer

6650-1 LG

Protocol version

HTTP1

VPC

vpc-0387714e4ef1306a7 Z

Total targets

2

Healthy

2

Unhealthy

0

Unused

0

Initial

0

Draining

0

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

Targets

Monitoring

Health checks

Attributes

Tags

Registered targets (2)

Filter targets

Deregister

Register targets

Instance ID

Name

Port

Zone

Health status

Health status details

I-9c4610e4d405842ae

8080

us-west-2a

Healthy

I-02967c21e8d1e6dd8

8080

us-west-2a

Healthy

Load Balancer

ALB forwards to the target group shown above

EC2 > Load balancers > 6650-1

6650-1

⌕ Actions

▼ Details

Load balancer type

Application

Status

Active

VPC

vpc-0387714e6ef1306a7

IP address type

IPv4

Scheme

Internet-facing

Hosted zone

Z1H1FLSHAB5F5

Availability Zones

subnet-09c52067saefd09e7 us-west-2b (usw2-az1)
subnet-0f25cfa36f491704 us-west-2a (usw2-az2)

Date created

November 10, 2023, 19:08 (UTC-08:00)

Load balancer ARN

arn:aws:elasticloadbalancing:us-west-2:241585107774:loadbalancer/app/6650-1/67f0b057e7ba81e8

DNS name

6650-1-1876966533.us-west-2.elb.amazonaws.com (A Record)

Listeners and rules

Network mappingSecurityMonitoringIntegrationsAttributesTags

Listeners and rules (1) Info

⌕ Manage rules Manage listener Add listener

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners

< 1 > ⌕

☐

Protocol:Port

☐

Default action

☐

Rules

☐

ARN

☐

Security policy

☐

Default SSL/TLS certificate

☐

Tags

Forward to target group

• 6650-1 1 (100%)
• Group-level stickiness: Off

1 rule

ARN

Not applicable

Not applicable

0 tags

Auto Scaling Group

ASG that manages the EC2 Instances

EC2 > Auto Scaling groups > 6650

6650

DetailsActivityAutomatic scalingInstance managementMonitoringInstance refresh

Group details

Auto Scaling group name

6650

Desired capacity

2

Status

-

Amazon Resource Name (ARN)

arn:aws:autoscaling:us-west-2:241585107774:autoScalingGroup:04c88a42-d4e1-4d5f-b243-0179950cdd2:autoScalingGroupName/6650

Date created

Fri Nov 10 2023 19:08:38 GMT-0800 (PST)

Minimum capacity

2

Maximum capacity

3

Launch template

Launch template

lt-087bae66a8784c863
6650

AMI ID

ami-00b7cc7d7a9f548ea

Instance type

t2.micro

Owner

arn:aws:sts:241585107774:assumed-role/voclabs/user2832489=liang.xim@northeastern.edu

Version

3

Security groups

-

Security group IDs

sg-0ac05408a8a5ab63a
sg-02dcffa100434b271
sg-0e7c8015c48d62185
sg-087da6511cc9e97ee
sg-030336f1426fe03b5

Create time

Sun Nov 12 2023 15:32:47 GMT-0800 (PST)

Description

-

Storage (volumes)

-

Key pair name

CS6650

Request Spot Instances

No

View details in the launch template console

RDS

Postgres DBs Running in RDS as the Storage Engine

Databases (2)

Group resources⌕ Modify Actions Restore from S3 Create database

Filter by databases

< 1 > ⌕

☐

DB identifier ▲

☐

Status ▼

☐

Role ▼

☐

Engine ▼

☐

Region & AZ ▼

☐

Size ▼

☐

Actions ▼

☐

CPU ▼

☐

Current activity ▼

☐

Maintenance ▼

☐

VPC ▼

☐

Multi-AZ ▼

database-1

Available

Instance

PostgreSQL

us-west-2d

db.t3.micro

2 Actions

3.41%

0.00 sessions

none

vpc-0387714e6ef1306a7

No

database-2

Available

Instance

PostgreSQL

us-west-2a

db.t4g.micro

2 Actions

3.09%

0.00 sessions

none

vpc-0387714e6ef1306a7

No