

distributed-systems-work

Assignments Repo for CS6650 Building Scalable Distributed Systems 23fall@neu

Ayan Mao 2023 Dec

<https://gortonator.github.io/bsds-6650/assignments-2022/Assignment-1>

<https://gortonator.github.io/bsds-6650/assignments-2022/Assignment-2>

<https://gortonator.github.io/bsds-6650/assignments-2022/Assignment-3>

Github link: <https://github.com/98may/distributed-systems-work/>

distributed-systems-work

1: Overview

 1.1: milestones

 1.2: client code design

 loadTester.java

 ApiTask.java

 Utilities.java

 Config.java

 1.3: server design with RabbitMQ

 1) add review servlet

 2) enable RabbitMQ

 1.4: key info

 AWS EC2

 server address

2: Database

 2.1: choice

 2.2: data model

 2.3: results

3: test results

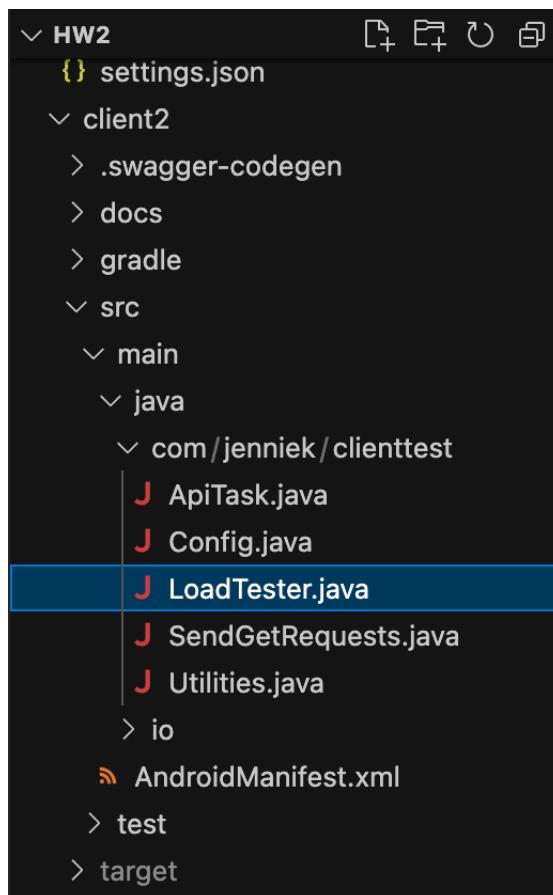
4: Appendix of Command

1: Overview

1.1: milestones

- 1) update Java server to support album review - like and dislike
- 2) update client to send posts (4 post per thread = 1 new album + 2 like + 1 dislike)
- 3) baseline tests without MQ
- 4) update server with RabbitMQ
- 5) real tests with RabbitMQ

1.2: client code design



loadTester.java

loadTester is the main java class to handle tests,

```
parse to set up args from Usage: LoadTester <threadGroupSize> <numThreadGroups> <delay>
[java|go],
```

and then mainLoad send out the major requests.

```
public static void main(String[] args) {
    LoadTester loadTester = new LoadTester();
    loadTester.parse(args);
    loadTester.mainLoad();
}
```

ApiTask.java

implements `Runnable`

as a single thread unit to send 4 post requests:

// 4 requests per thread: POST a new album and image + POST two likes and one dislike for the album. (4 = 1 album + 2 like + 1dislike)

Utilities.java

mainly the Utility functions for logging and numbers

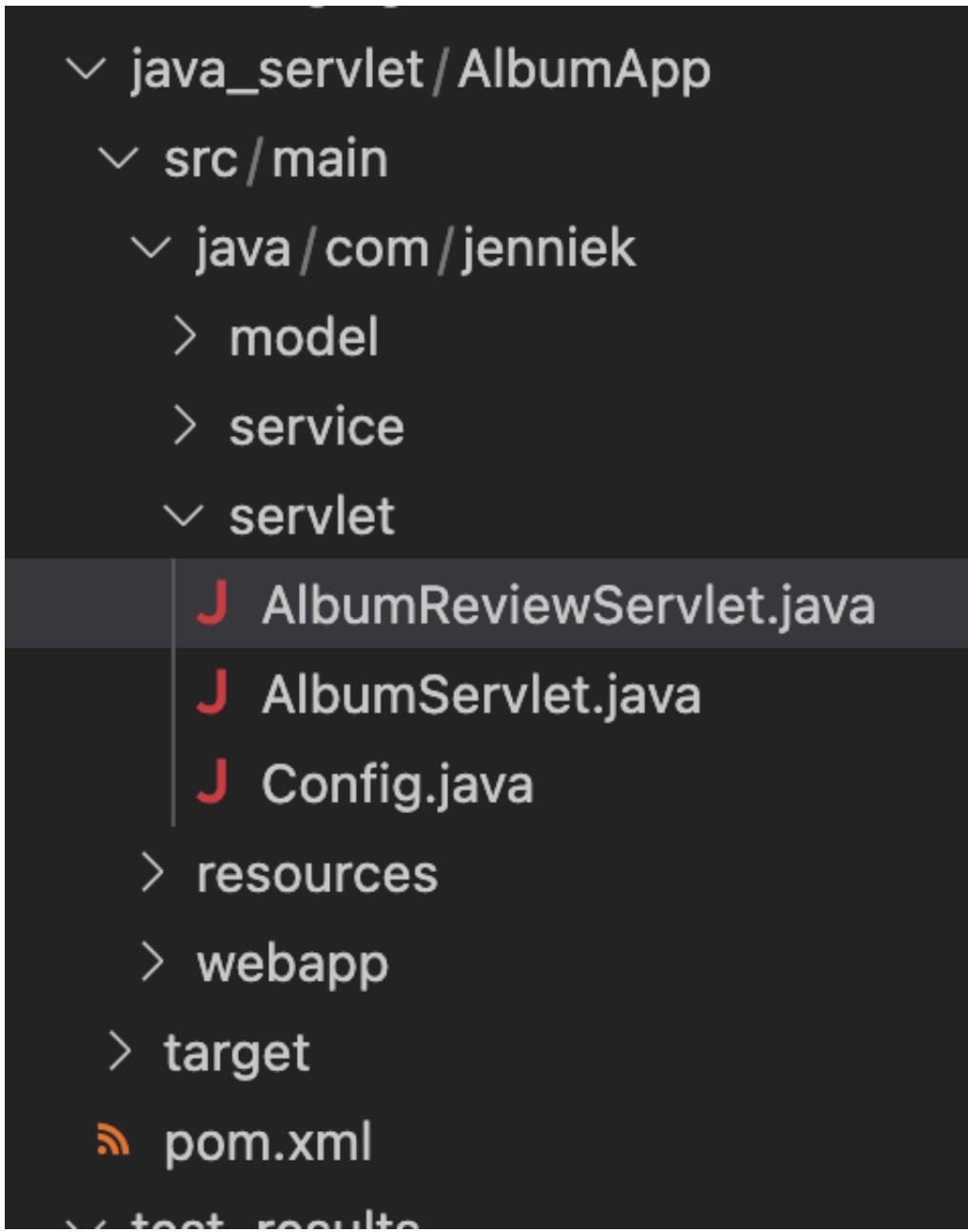
Config.java

set up the public static final arguments like `INITIAL_THREAD_COUNT`, `CLIENT_LOG_PATH` and `javaServletAddress`

Change this from 1000 to 100 `public static final int LOAD_TEST_REQUESTS_PER_THREAD = 100;` as this hw3 says "To reduce the new album write load, Just write 100 new albums per thread iteration instead of 1000".

1.3: server design with RabbitMQ

1) add review servlet



```
// AlbumReviewServlet.java
@WebServlet("/review/*")
public class AlbumReviewServlet extends HttpServlet {
    // implementation for handling like/dislike

// AlbumServlet.java
@WebServlet("/albums/*")
@MultipartConfig
public class AlbumServlet extends HttpServlet {
```

2) enable RabbitMQ

use producer and consumer

The screenshot shows a CloudShell session with several browser tabs open. The tabs include:

- CS6650 Fall 2023 (49 unread)
- albumMQ | Brokers | Amazon
- CS6650 Spring 2023 Assignm...
- Dashboard
- ChatGPT - distributed system...
- 3.80.33.155.8080/AlbumApp
- Your Orders

The main content area displays the AWS Management Console for Amazon MQ. It shows the details of a broker named "albumMQ".

Details

Specifications	Configuration	Security and network	Maintenance
ARN Info arnaws:mq:us-east-1:211832168273:broker:albumMQ:b-81f3704c-bd80-4c81-a663-8edf8828c872	Configuration name Info albumMQ-configuration	VPC Info vpc-071fd8305cf646176	Automatic minor version upgrade Yes
Broker name Info albumMQ	Configuration revision Revision 1 - Auto-generated default for albumMQ-configuration on RabbitMQ 3.11.20	Subnet(s) Info subnet-0bcb08583efadfc683	Maintenance window Wednesday 17:00 - 19:00 UTC
Broker status Info Running	CloudWatch Logs General Disabled - Logs	Public accessibility Info Yes	
Broker instance type Info mq.t3.micro			
Deployment mode Info Single-instance broker			
Broker engine Info RabbitMQ			
Broker engine version 3.11.20			
Creation time (Local) December 3, 2023 at 21:42 (UTC-8:00)			

Connections

Access your queues and exchanges and connect your application to the broker. If you disable public accessibility for your broker, your endpoints are reachable only within a VPC.

RabbitMQ web console

Step 3: Configure settings

Step 4: Review and create

Broker name: albumMQ

Deployment mode: Single-instance broker

Estimated deployment time: 20 minutes

Broker instance type: mq.t3.micro

Storage type: Amazon Elastic Block Store

Broker engine: RabbitMQ

RabbitMQ access

The credentials used to access your broker via the RabbitMQ web console.

Username: ds-db

Password: 012345678910

Additional settings

Tags - optional

No tags associated with the resource.

Add new tag

1.4: key info

AWS EC2

delete load balancer from hw2, so only use one AWS EC2

EC2 > Instances > i-0d3a7ef8476b4c5f0

Instance summary for i-0d3a7ef8476b4c5f0 (ds_server0) [Info](#)

Updated less than a minute ago

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0d3a7ef8476b4c5f0 (ds_server0)	3.80.33.155 [open address]	172.31.21.99
IPv6 address	Instance state	Public IPv4 DNS
-	Running	ec2-3-80-33-155.compute-1.amazonaws.com [open address]
Hostname type	Private IP DNS name (IPv4 only)	Elastic IP addresses
IP name: ip-172-31-21-99.ec2.internal	ip-172-31-21-99.ec2.internal	-
Answer private resource DNS name	Instance type	AWS Compute Optimizer finding
IPV4 (A)	t2g.nano	Opt-in to AWS Compute Optimizer for recommendations. Learn more [?]
Auto-assigned IP address	VPC ID	Auto Scaling Group name
3.80.33.155 [Public IP]	vpc-071fd8305cf646176	-
IAM Role	Subnet ID	
-	subnet-0bcb08583efad683	
IMDSv2		
Required		

Details Security Networking Storage Status checks Monitoring Tags

Instance details [Info](#)

Platform	AMI ID	Monitoring
Amazon Linux (Inferred)	ami-04a3fea0ceec717e5	disabled
Platform details	AMI name	Termination protection
Linux/UNIX	al2023-ami-2023.2.20231002.0-kernel-6.1-arm64	Disabled
Stop protection	Launch time	AMI location
Disabled	Mon Oct 09 2023 03:33:00 GMT-0700 (Pacific Daylight Time) (about 2 months)	amazon/al2023-ami-2023.2.20231002.0-kernel-6.1-arm64
Instance auto-recovery	Lifecycle	Stop-hibernate behavior
Default	normal	Disabled
AMI Launch index	Key pair assigned at launch	State transition reason
0	ec2_2	-
Credit specification	Kernel ID	State transition message
unlimited	-	-

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

server address

```
public static final String javaServletAddress =
"http://3.80.33.155:8080/AlbumApp";
```

2: Database

2.1: choice

AWS RDS + MySQL 8.0.33

2.2: data model

alter the albums table in hw2 and clear it in the beginning for better performance

table: albums

field: album_id (PK), name, artist, release_year, image, likes, dislikes

```
MySQL [ds_hw2_db]> describe albums;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| album_id | varchar(50) | NO | PRI | NULL |       |
| name | varchar(100) | NO |       | NULL |       |
| artist | varchar(100) | NO |       | NULL |       |
| release_year | varchar(4) | YES |       | NULL |       |
| image | varchar(255) | YES |       | NULL |       |
| likes | int | YES |       | 0 |       |
| dislikes | int | YES |       | 0 |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.058 sec)
```

init albums table example:

```
MySQL [ds_hw2_db]> select * from albums;
+-----+-----+-----+-----+-----+-----+
| album_id | name | artist | release_year | image | likes | dislikes |
+-----+-----+-----+-----+-----+-----+
| 1 | offers | Ayan | 2023 | NULL | 0 | 0 |
| 2 | offers | Ayan | 2023 | nmtb.jpg | 100 | 0 |
| 28ef6e80-9259-11ee-8d93-16914fa7a575 | 1 | Unknown Artist | 1998 | NULL | 0 | 0 |
| 3 | offers3 | Ayan3 | 2023 | nmtb.jpg | 300 | 1 |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.001 sec)
```

2.3: results

simple postman test succeeded:

The screenshot shows a dual-monitor setup. The left monitor displays the Postman application interface, which includes a sidebar with collections like 'AlbumReview' and 'test_album_go', and a main panel for a 'AlbumReview / like' request. The right monitor displays a terminal window with MySQL command-line output. The terminal shows the creation of a database, the creation of an 'albums' table with specific columns and constraints, and the insertion of four sample records into the table. The MySQL session ends with a 'quit' command.

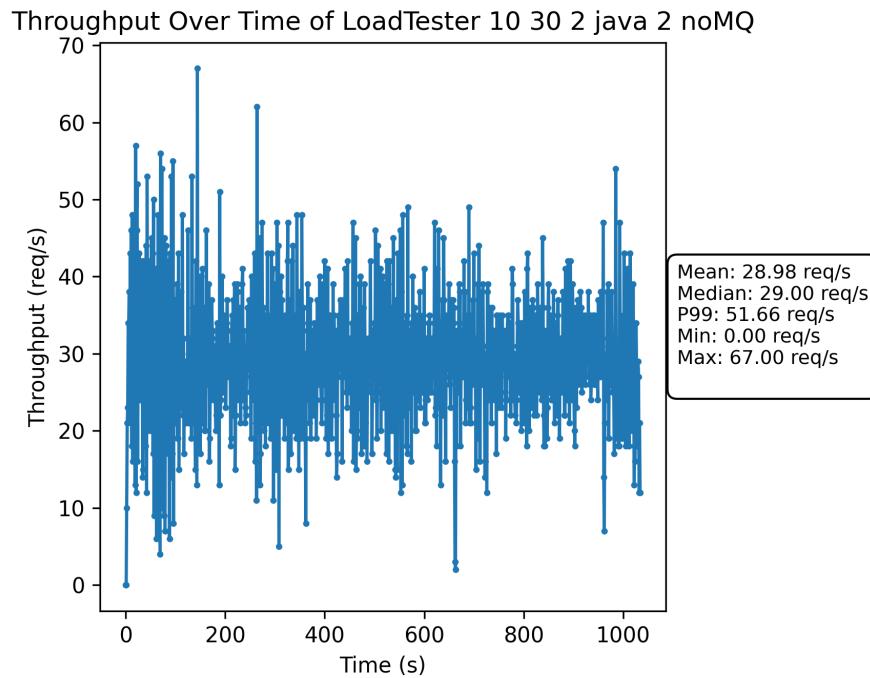
```
mysql -h dadatabase.ccfcharlh91s.us-east-1.rds.amazonaws.com -u 01234567 -p
show databases;
use ds_hw2_db;
describe albums;
select * from albums;
```

As you can see in the graph, it could have hundreds of thousands of records in the albums table.

3: test results

baseline test result without MQ

```
---  
Executing: LoadTester 10 30 2 java  
  
===== Start hw3 Main Load Phase =====  
2023-12-03 22:18:59  
All tasks completed.  
totalRequests: 120000, Failure Rate: 0.0  
failedRequests: 0  
successfulRequests: 120000  
2023-12-03 22:18:59 - 2023-12-03 22:36:13  
Wall Time: 1034 seconds  
Throughput: 116 requests/second  
----- POST ALBUM REVIEW request statistics -----  
Mean: 2985.3660666666665 ms  
Median: 3062.0 ms  
P99: 4875.0 ms  
Min: 93 ms  
Max: 7560 ms  
----- POST ALBUM request statistics -----  
Mean: 801.6368666666667 ms  
Median: 852.0 ms  
P99: 1361.0 ms  
Min: 90 ms  
Max: 2946 ms  
2023-12-03 22:36:13  
===== End hw3 Main Load Phase =====
```



real test result with RabbitMQ enabled

```
(base) may@ayanmaos-m1-mbp client % mvn clean install && mvn exec:java -Dexec.mainClass="com.jenniek.clienttest.LoadTester" -Dexec.args="10 30 2 java"
```

4: Appendix of Command

Please login as the user "ec2-user" rather than the user "root".

local

```
// ssh to aws ec2
```

```

/Users/may/Desktop/neu/cs6650_distributed/shortcuts/ssh_ec2.sh

// chmod 400 ec2_2.pem
// ssh -i ec2_2.pem ec2-user@ec2-3-80-33-155.compute-1.amazonaws.com
ssh -i ec2_2.pem ec2-user@ec2-54-221-189-103.compute-1.amazonaws.com

// mysql to aws rdb
mysql -h dsdatabase.ccfcharlh91s.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
// pwd: 01234567

/* java servlet */
// local build java servlet in Maven project to generate the AlbumApp.war file
mvn clean install
// scp java servlet - AlbumApp.war to aws ec2
scp -i /Users/may/Downloads/ec2_2.pem
/Users/may/Desktop/neu/cs6650_distributed/distributed-systems-
work/hw3/java_servlet/AlbumApp/target/AlbumApp.war ec2-user@ec2-3-80-33-
155.compute-1.amazonaws.com:~/tmp

/* Go server */
// local run go server(macOS + m1 chip) for local tests
GOOS=darwin GOARCH=arm64 go run ./main.go
// local build go server for aws ec2(linux + arm64)
GOOS=linux GOARCH=arm64 go build -o server main.go
// scp go server - server to ec2 ~/emp to start
scp -i /Users/may/Downloads/ec2_2.pem
/Users/may/Desktop/neu/cs6650_distributed/distributed-systems-
work/hw1/go_server/go-server-generated/server ec2-user@ec2-3-80-33-
155.compute-1.amazonaws.com:~/tmp

/* client - LoadTester */
`Usage: LoadTester <threadGroupSize> <numThreadGroups> <delay> [java|go]` 
// e.g.

mvn clean install && mvn exec:java -
Dexec.mainClass="com.jenniek.clienttest.LoadTester" -Dexec.args="10 30 2 java"

mvn exec:java -Dexec.mainClass="com.jenniek.clienttest.LoadTester" -Dexec.args="10
30 2 java"

// plot throughput
conda activate base
/Users/may/anaconda3/bin/python

```

```
/Users/may/Desktop/neu/cs6650_distributed/distributed-systems-work/hw2/test_results/plot.py
```

aws ec2

```
// java servlet
sudo mv ~/tmp/AlbumApp.war /usr/share/tomcat/webapps && sudo systemctl restart tomcat && systemctl status tomcat

sudo mv ~/tmp/AlbumApp.war /usr/share/tomcat/webapps

sudo systemctl restart tomcat
sudo systemctl stop tomcat
systemctl status tomcat

sudo vim /usr/share/tomcat/logs/catalina.out
sudo tail /usr/share/tomcat/logs/catalina.out -n 200

// see log to debug java servlet
sudo bash -c 'echo > /usr/share/tomcat/logs/catalina.out'
sudo grep "album" /usr/share/tomcat/logs/catalina.out

// go server
~/tmp/server

// mysql to aws rdb
mysql -h dsdatabase.ccfcharlh91s.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
    // 01234567
show databases;
use ds_hw2_db;
describe albums;
select COUNT(*) from albums;
SELECT * FROM albums ORDER BY RAND() LIMIT 20;

ALTER TABLE albums
ADD COLUMN likes INT DEFAULT 0,
ADD COLUMN dislikes INT DEFAULT 0;

INSERT INTO albums (album_id, name, artist, release_year, image, likes, dislikes)
VALUES ('1', 'offers', 'Ayan', 2023, NULL, 0, 0);
```

```
INSERT INTO albums (album_id, name, artist, release_year, image, likes, dislikes)
VALUES ('2', 'offers', 'Ayan', 2023, 'nmtb.jpg', 100, 0);
```

```
INSERT INTO albums (album_id, name, artist, release_year, image, likes, dislikes)
VALUES ('3', 'offers3', 'Ayan3', 2023, 'nmtb.jpg', 300, 1);
```

```
INSERT INTO albums (album_id, name, artist, release_year, image, likes, dislikes)
VALUES ('1230', 'offers1230', 'Ayan1230', 1998, 'nmtb.jpg', 12300, 50);
```