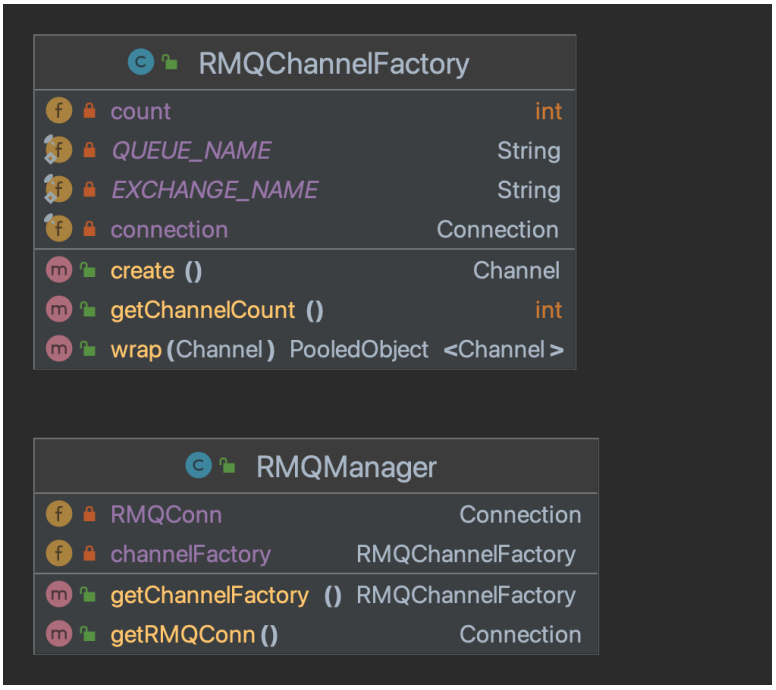
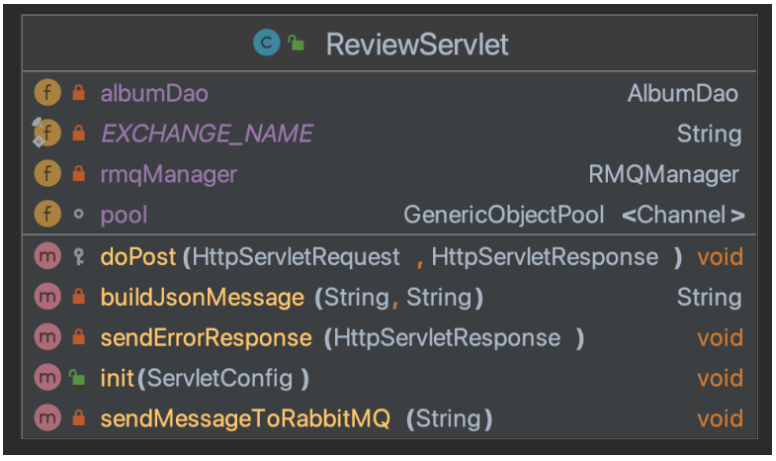
Assignment3 Report

https://github.com/peihsuan-lin/CS6650/tree/main/assignment3

Server Design





Major Classes

1. **RMQChannelFactory.java**

   This class is a custom factory for creating and managing a pool of reusable RabbitMQ `Channel` objects.

2. **RMQManager.java**

   Manages the overall RabbitMQ connection and interacts with the `RMQChannelFactory` for channel pooling.

3. **ReviewServlet.java**

   This servlet is a part of the Album Store application, handling HTTP requests related to reviews. It interacts with RabbitMQ, uses Gson for JSON processing, and interfaces with data access objects (DAOs). It's responsible for handling web requests, processing them, and sending review messages to a RabbitMQ server.

Message Flow Logic

- **Creation**: When a channel is needed, the `RMQChannelFactory` creates one.
- **Pooling**: After creation, channels are managed by the Apache Commons `GenericObjectPool`. The pool ensures efficient reuse of these RabbitMQ channels.
- **Usage**: `RMQManager` (initiated in `ReviewServlet`) would request channels from `RMQChannelFactory`. After use, these channels are returned to the pool.

Pool Settings

- **Optimized Pool Size:** I have experimented with different pool sizes (range 80 ~ 250) and found that setting the pool size to 200 results in a balance between resource allocation and system performance, achieving a throughput of 1000 requests per second.
- **Concurrent Thread Groups**: The consumer thread groups are configured to 30, to evaluate how well the system maintains performance when facing a surge in message traffic.

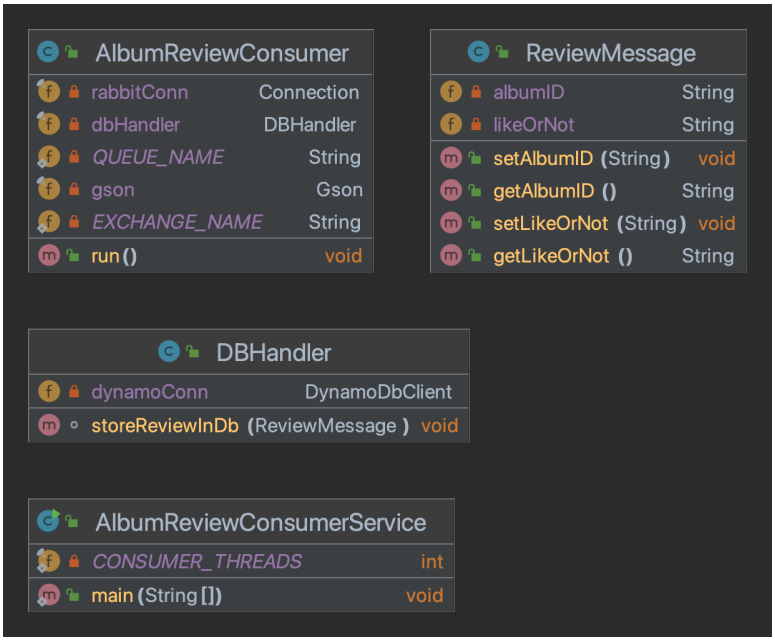
```
Test load:
threadGroupSize: 10, numThreadGroups: 30, delay: 2
Time taken: 31050 ms
Number of successful requests: 30999
Number of fail requests: 1
Walltime: 30.992 seconds
Total throughput: 1000.2581311306143 req/s

Metrics for POST_ALBUM:
Mean response time: 161.84125294364335 ms
Median response time: 142.0 ms
99th response time: 448.00200000000007 ms
Min response time: 36.0 ms
Max response time: 1136.0 ms

Metrics for POST_REVIEW:
Mean response time: 97.48584461867426 ms
Median response time: 64.0 ms
99th response time: 332.02600000000024 ms
Min response time: 15.0 ms
Max response time: 954.0 ms
```

Consumer Design

Major Classes



1. **AlbumReviewConsumer.java**

   Implements the `Runnable` interface for processing messages from the RabbitMQ queue. It takes a channel from the channel pool and uses it to consume review

messages. Once a message is received, the `AlbumReviewConsumer` will deserialize and utilized `DBHandler` to interact with database.

2. **AlbumReviewConsumerService.java**

   This class is responsible for initializing and managing a multi-threaded environment for message consumption.

3. **DBHandler.java**

   Handles database interactions and establishes a connection to DynamoDB. It provides the method `storeReviewInDb`, which is responsible for persisting message content into the database. Upon invocation, this method performs the following steps:

   - **Receiving a `ReviewMessage`**: Initially, it accepts a `ReviewMessage` object, the data received from a RabbitMQ message.
   - **Transformation**: It then transforms this `ReviewMessage` into a format suitable for DynamoDB storage.
   - **Persistence**: Finally, the method interacts with the DynamoDB client to store the transformed data.

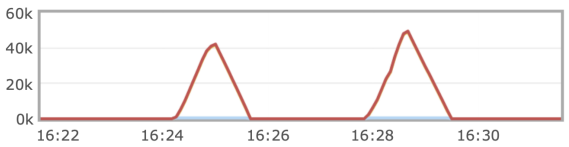
4. **ReviewMessage.java**

   A simple POJO (Plain Old Java Object) for managing review `albumID` and `likeOrNot` Fields of message data.

Queue Consumers Threads

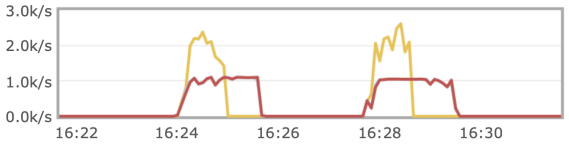
After extensive testing, it was determined that **250 consumer threads** provide the optimal balance for the system. The message rate resemble a trapezoid shape, indicates that the consumer is able to process messages in bursts, effectively clearing the queue before it builds up again, thus preventing the queue from growing indefinitely.

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



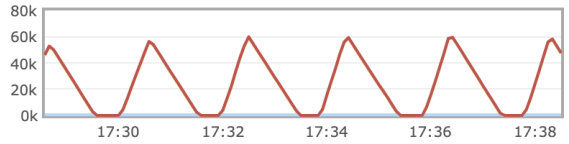
Ready	0 msg
Unacknowledged	0 msg
Total	0 msg

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



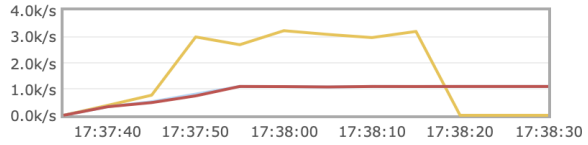
Publish	0.00/s
Deliver	0.00/s
Acknowledge	0.00/s

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



Ready	41,867 msg
Unacknowledged	500 msg
Total	42,367 msg

Message rates (chart: last minute) (?)



Publish	0.00/s
Deliver	1,100/s
Acknowledge	1,100/s