Report for Assignment3

Git Repo

Please visit **Github Repo**

Design

- Database Design
  - For Skier Microservice, the database stored data by `key = skierID, value={seasonID, dayID, vertical=liftID*10, liftID}`.
  - For Resort Microservice, the database stored data by `key = day+dayID, value = {seasonID, resortID, skierID, liftID, time}`.
- Deployment on AWS
  - Server
    - Build artifacts of Server, then scp into server's tomcat webapp folder.
  - Client
    - Same as **Assignment2**.
  - Consumer
    - Switch into Consumer folder, run `zsh(or bash) build.sh` which will build the current image and push that into `zjdx1998/consumer` with latest tag.
    - For your convenience, run

```
docker pull zjdx1998/consumer:mar11-amd64-latest
docker run -it --rm --network="host" --name consumer
zjdx1998/consumer:mar11-amd64-latest localhost skier
```

to start skier micro service.

And run

```
docker pull zjdx1998/consumer:mar11-amd64-latest
docker run -it --rm --network="host" --name consumer
zjdx1998/consumer:mar11-amd64-latest localhost resort
```

to start resort micro service.

Here is a screenshot of skier microservice running in aws ec2.

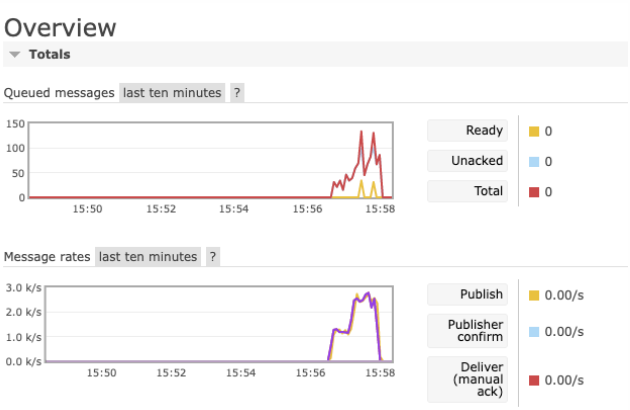
Test Runs

Step1

For 128 threads, the client arguments are: `-nt 128 -ns 20000 -nl 40 -nr 10 -server PUBLIC_IP_ADDRESS`

Start the microservice by `docker run -it --rm --network="host" --name consumer zjdx1998/consumer:mar11-amd64-latest localhost skier`.

```
/Users/jeromy/.sdkman/candidates/java/17-open/bin/java ...
Success
Ready to run phases!
Phase1 is ready to start!
Phase1 should execute 32 threads with 1250 requests each.
Phase1 has already completed 20.0% tasks
Phase2 is ready to start!
Phase2 should execute 128 threads with 937 requests each.
Phase2 has already completed 20.0% tasks
Phase3 is ready to start!
Phase3 should execute 12 threads with 1 requests each.
Phase3 has already completed 100.0% tasks
Number of Successful Requests Sent: 154710
Number of Unsuccessful Requests: 0
Total run time: 81383 (ms)
Total Throughput in requests per second: 1901.0112677094726
Mean response time: 46.66664729958296
Median response time: 37.0
Throughput: 21.42858032162376
99th response time: 191.0
min and max response time: min: 11.0 , max: 4921.0

Process finished with exit code 0
```



For 256 threads, the client arguments are: `-nt 256 -ns 20000 -nl 40 -nr 10 -server PUBLIC_IP_ADDRESS`

Start the microservice by `docker run -it --rm --network="host" --name consumer zjdx1998/consumer:mar11-amd64-latest localhost skier`.

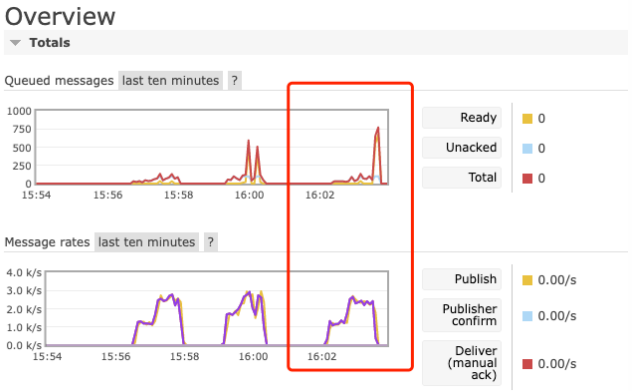
```
/Users/jeromy/.sdkman/candidates/java/17-open/bin/java ...
Success
Ready to run phases!
Phase1 is ready to start!
Phase1 should execute 64 threads with 625 requests each.
Phase1 has already completed 20.0% tasks
Phase2 is ready to start!
Phase2 should execute 256 threads with 468 requests each.
Phase2 has already completed 20.0% tasks
Phase3 is ready to start!
Phase3 should execute 25 threads with 1 requests each.
Phase3 has already completed 100.0% tasks
Number of Successful Requests Sent: 147824
Number of Unsuccessful Requests: 0
Total run time: 67560 (ms)
Total Throughput in requests per second: 2188.040260509177
Mean response time: 86.33644847182522
Median response time: 62.0
Throughput: 11.582593651930644
99th response time: 382.0
min and max response time: min: 13.0 , max: 6628.0

Process finished with exit code 0
```

Step2

For 128 threads, the client arguments are: `-nt 128 -ns 20000 -nl 40 -nr 10 -server PUBLIC_IP_ADDRESS`

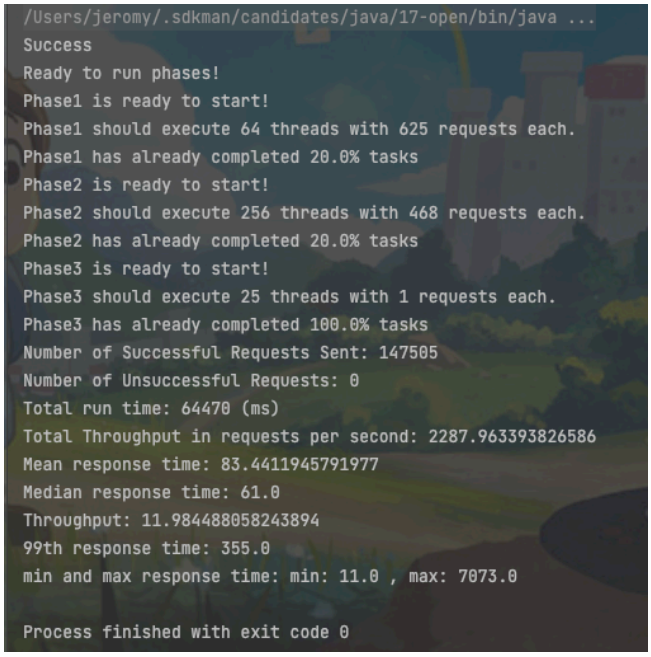
Start the microservice by `docker run -it --rm --network="host" --name consumer zjdx1998/consumer:mar11-amd64-latest localhost resort` .

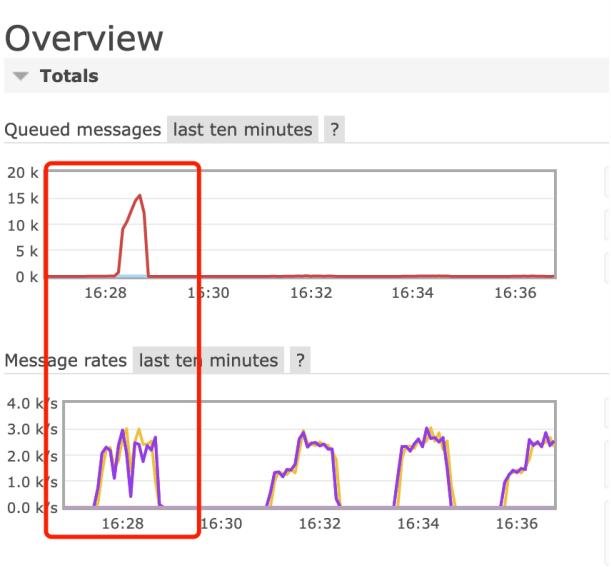
```
/Users/jeromy/.sdkman/candidates/java/17-open/bin/java ...
Success
Ready to run phases!
Phase1 is ready to start!
Phase1 should execute 32 threads with 1250 requests each.
Phase1 has already completed 20.0% tasks
Phase2 is ready to start!
Phase2 should execute 128 threads with 937 requests each.
Phase2 has already completed 20.0% tasks
Phase3 is ready to start!
Phase3 should execute 12 threads with 1 requests each.
Phase3 has already completed 100.0% tasks
Number of Successful Requests Sent: 156182
Number of Unsuccessful Requests: 0
Total run time: 81966 (ms)
Total Throughput in requests per second: 1905.4486006392895
Mean response time: 46.733741436548904
Median response time: 38.0
Throughput: 21.397815994632378
99th response time: 226.0
min and max response time: min: 11.0 , max: 1210.0

Process finished with exit code 0
```



For 256 threads, the client arguments are: `-nt 256 -ns 20000 -nl 40 -nr 10 -server PUBLIC_IP_ADDRESS`

Start the microservice by `docker run -it --rm --network="host" --name consumer zjdx1998/consumer:mar11-amd64-latest localhost resort`.





Step Combined

For 128 threads, the client arguments are: `-nt 128 -ns 20000 -nl 40 -nr 10 -server PUBLIC_IP_ADDRESS`

Start the microservice by `docker run -it --rm --network="host" --name consumer1 zjdx1998/consumer:mar11-amd64-latest localhost resort && docker run -it --rm --network="host" --name consumer2 zjdx1998/consumer:mar11-amd64-latest localhost skier`.

```
/Users/jeromy/.sdkman/candidates/java/17-open/bin/java ...
Success
Ready to run phases!
Phase1 is ready to start!
Phase1 should execute 32 threads with 1250 requests each.
Phase1 has already completed 20.0% tasks
Phase2 is ready to start!
Phase2 should execute 128 threads with 937 requests each.
Phase2 has already completed 20.0% tasks
Phase3 is ready to start!
Phase3 should execute 12 threads with 1 requests each.
Phase3 has already completed 100.0% tasks
Number of Successful Requests Sent: 156062
Number of Unsuccessful Requests: 0
Total run time: 76663 (ms)
Total Throughput in requests per second: 2035.6886633708568
Mean response time: 44.98519784207192
Median response time: 36.0
Throughput: 22.22953433506434
99th response time: 229.0
min and max response time: min: 11.0 , max: 1608.0

Process finished with exit code 0
```

For 256 threads, the client arguments are: `-nt 256 -ns 20000 -nl 40 -nr 10 -server PUBLIC_IP_ADDRESS`

Start the microservice by `docker run -it --rm --network="host" --name consumer1 zjdx1998/consumer:mar11-amd64-latest localhost resort && docker run -it --rm --network="host" --name consumer2 zjdx1998/consumer:mar11-amd64-latest localhost skier`.

```
/Users/jeromy/.sdkman/candidates/java/17-open/bin/java ...
Success
Ready to run phases!
Phase1 is ready to start!
Phase1 should execute 64 threads with 625 requests each.
Phase1 has already completed 20.0% tasks
Phase2 is ready to start!
Phase2 should execute 256 threads with 468 requests each.
Phase2 has already completed 20.0% tasks
Phase3 is ready to start!
Phase3 should execute 25 threads with 1 requests each.
Phase3 has already completed 100.0% tasks
Number of Successful Requests Sent: 151443
Number of Unsuccessful Requests: 0
Total run time: 61188 (ms)
Total Throughput in requests per second: 2475.0441262992745
Mean response time: 81.14130191092883
Median response time: 57.0
Throughput: 12.324179874482777
99th response time: 538.0
min and max response time: min: 12.0 , max: 4613.0

Process finished with exit code 0
```

Mitigation Strategy

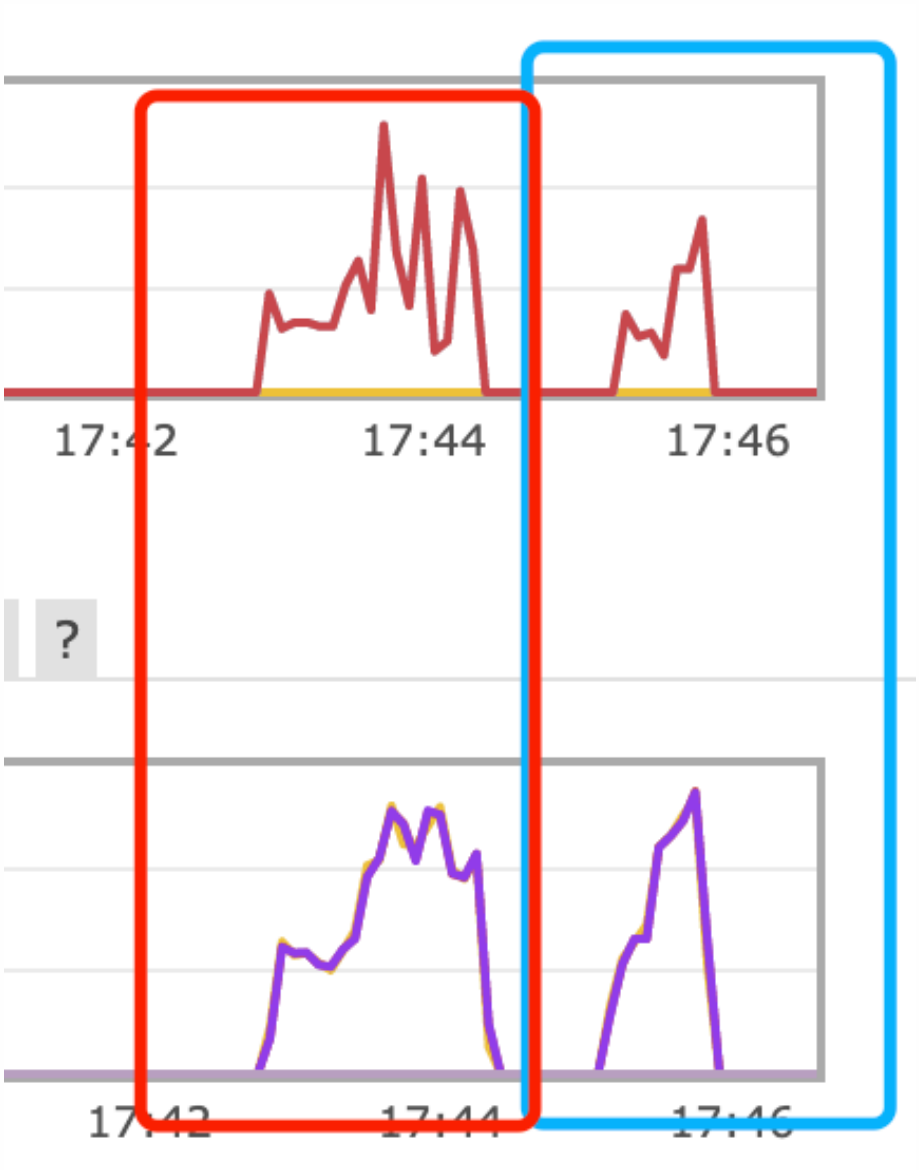
I believe it's totally okay if we don't use any mitigation strategy because the number of queued messages is less than 100 which is quite small.

But I still deploy the `EventCountCircuitBreaker` into Server.

The Server use `breaker = new EventCountCircuitBreaker(500, 5, TimeUnit.SECONDS, 300);` which means check every 5 seconds to see if the current requests are more than 500 and stop processing requests(i.e. send to rabbitmq) if so, and resume processing when the request per 5 seconds is smaller than 300.

From the below chart, we can see a relatively noticeably improvement on queued messages for 256 client threads.

Meanwhile, I reused the load balancer from last assignment, and the results are:



That's a huge improvement compared to the previous one. But the network stability issues should also be considered into the factors.

Thanks for your reading!